La synchronisabilité pour les systèmes distribués

Laetitia Laversa

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

La synchronisabilité pour les systèmes distribués

Laetitia LAVERSA

Laboratoire d'Informatique, de Signaux et Systèmes de Sophia Antipolis (I3S)
UMR7271 UCA CNRS

Présentée en vue de l'obtention du grade de docteur en INFORMATIQUE **d'**Université Côte d'Azur

Dirigée par : Étienne LOZES,
Professeur des Universités,
Université Côte d'Azur

Co-encadrée par : Cinzia DI GIUSTO,
Maîtresse de Conférences,
Université Côte d'Azur

Soutenue le : 14 décembre 2021

Devant le jury, composé de :

Julien LANGE,
Lecturer,
Royal Holloway, University of London

Philippe QUEINNEC,
Professeur des Universités,
Institut National Polytechnique de Toulouse

Gwen SALAÜN,
Professeur des Universités,
Université Grenoble Alpes

Fabrice HUET,
Professeur des Universités,
Université Côte d'Azur

LA SYNCHRONISABILITÉ POUR LES SYSTÈMES DISTRIBUÉS

Synchronizability for distributed systems

Laetitia LAVERSA

Jury :

Rapporteurs

Julien LANGE,
Lecturer,
Royal Holloway, University of London

Philippe QUEINNEC,
Professeur des Universités,
Institut National Polytechnique de Toulouse

Gwen SALAÜN,
Professeur des Universités,
Université Grenoble Alpes

Examinateur

Fabrice HUET,
Professeur des Universités,
Université Côte d'Azur

Directeur de thèse

Étienne LOZES,
Professeur des Universités,
Université Côte d'Azur

Co-encadrant de thèse

Cinzia DI GIUSTO,
Maîtresse de Conférences,
Université Côte d'Azur

Université Côte d'Azur

Laetitia LAVERSA

La synchronisabilité pour les systèmes distribués

xiii+164 p.

Ce document a été préparé avec LaTeX2e et la classe `these-ISSS` version v. 2.10.

Impression : sommaire.tex – 22/12/2021 – 10:01

Révision pour la classe : these-ISSS.cls,v 2.10 2020/06/24 14:16:37 mpelleau

La synchronisabilité pour les systèmes distribués

Résumé

Les systèmes distribués sont omniprésents, cependant leur implémentation est complexe et sujette aux erreurs. Pour les vérifier, ils peuvent être modélisés en système d'automates communicants, où chaque automate représente le comportement d'un élément du système. Les propriétés liées à la vérification, telles que le problème de l'accessibilité, restent indécidables dans un tel modèle. En effet, un système d'automates communiquant de façon asynchrone est Turing équivalent. C'est pourquoi l'utilisation d'approximations est donc nécessaire. La k-synchronisabilité est l'une de ces approximations. Un système est k-synchronisable si pour toute exécution, il existe une exécution équivalente pouvant être divisée en phases, appelées k-échanges, contenant k messages. Dans cette thèse, nous nous intéressons à l'analyse des systèmes k-synchronisables : nous montrons que le problème de l'accessibilité est décidable et nous nous interrogeons sur la k-synchronisabilité d'un système donné. Pour ce deuxième point, nous regardons le cas où un k est passé en paramètre, et le cas où il faut trouver le k tel que le système est k-synchronisable. Nous prouvons que ces deux problèmes sont décidables lorsque le système communique en boîte aux lettres. Le premier problème est également décidable en pair-à-pair, mais, avec ce type de communication, le deuxième problème reste ouvert. Nous complétons cette étude en identifiant certaines implications contre-intuitives de la notion de k-synchronisabilité, celles-ci nous ont poussés à définir des variantes de la k-synchronisabilité. Une étude comparative de différentes classes de systèmes, soit étudiées ou définies dans cette thèse, soit provenant de la littérature, conclut nos travaux.

Mots-clés : Automates communicants, Synchronisabilité, Diagrammes de séquences

Synchronizability for distributed systems

Abstract

Distributed systems are ubiquitous and their implementation is complex and error-prone. In order to check for errors, they can be modeled as systems of communicating automata, where each automaton represents the behavior of an element of the system. Verification problems such as reachability are undecidable in such a model. Indeed, a system of communicating automata is Turing-equivalent. For that, the use of approximations is necessary. k-synchronizability is one of these techniques. A system is k-synchronizable if, for all execution, there is an equivalent execution that can be divided into phases containing k messages. These phases are called k-exchanges. In this thesis, we analyse k-synchronizable systems: we show that reachability is decidable and we are interested in the membership problem, that is: given a system, decide whether it is k-synchronizable. We study both the case where k is an input of the problem, and the case where we have to guess a k such that the system is k-synchronizable. We study them according to the type of communication of the system. Either the system is in mailbox and so each process has only one buffer to store all received messages, or the system is in peer-to-peer, and each process has a buffer for each sender. Both problems are decidable when the system is communicating in mailbox. When the system is communicating in peer-to-peer, the first one is decidable and the second remains open. Finally, we point out some counter-intuitive cases of k-synchronizable systems that lead us to propose some variations to the definition of k-synchronizability. A comparative study of the state-of-art classes of systems and our new classes concludes our work.

Keywords: Communicating automata, Synchronizability, Message sequence charts

Remerciements

Je remercie tout d'abord Philippe QUEINNEC, Julien LANGE et Gwen SALAÜN d'avoir accepté d'être rapporteurs de cette thèse, ainsi que Fabrice HUET, pour avoir accepté de faire partie du jury.

Je me dois de remercier mes directeurs de thèse qui m'ont fait confiance durant ces trois années. Merci à vous deux pour votre patience, votre enthousiasme et votre disponibilité. Cette thèse aurait été bien différente sans vous. Merci Étienne pour ta gentillesse, ton soutien, ta sollicitude et ton œil aiguisé pour les contre-exemples. Merci Cinzia pour tes encouragements, merci pour ton aide et pour tout le reste ; notre complicité m'était indispensable. Merci encore.

Durant ma deuxième année de thèse, j'ai eu la chance de pouvoir collaborer avec Alain FINKEL et Benedikt BOLLIG au LSV sur le plateau de Paris-Saclay. Merci à eux d'avoir accepté cette collaboration et de m'avoir permis d'accéder à un point de vue différent sur ma thèse. Thank you Amrita for our thinking sessions and Igor for your welcoming. Merci également à Amélie de m'avoir gentiment reçue dans son bureau.

Dès mon stage de M1 au Laboratoire I3S, j'ai eu la chance d'être accueillie par une équipe de doctorants chaleureux. En premier lieu, Ingrid et Ophélie, merci à vous de m'avoir fait sentir chez moi. Mais également Assia, Benjamin, Emilien, Heytem et Jonathan.

Merci aux permanents de mon équipe d'adoption pour leurs accompagnements et conseils et les réunions d'équipes essentielles : Adrien, Arnaud, Bruno, Elisabetta, Enrico, Jean-Charles, Joëlle, Sandrine et, tout particulièrement, Marie pour les templates, les chocolats, et ton investissement dans tout le reste.

Bien entendu, merci aux doctorants avec qui j'ai passé toutes ces années, ou une partie pour certains. Les gâteaux, le soutien et l'aide constant de Rémy, merci d'avoir répondu à toutes les questions de la Terre ; l'aide et le soutien made in Italy de Sara, grazie mille per esserci e per essere te (e grazie Pietro per le pizze !) ; l'humour approximatif de Laetitia (Grasse RPZ) ; la connivence avec le frère d'encadrants Loïc. Merci à tous les autres d'avoir été là, et d'avoir donné un peu de vous dans cette équipe de cœur : Amélie, Arthur, Diana, François, Giulia, Piotr, Samvel, ainsi que les derniers arrivés Alexandre, Aymeric, Florian, Nina, Romain, Steve et Victor.

Merci à mes compagnons de route de longue date, Nico, cher co-bureau, et Adrien. C'est bon, on l'a fait ! Un merci également à Claire et Dayle grâce à qui notre aventure MT180 a pris une autre dimension.

Merci à mes amis, ici et souvent ailleurs, Charlyne, Jérém, Morgane et Roxane, pour les apartés nécessaires et revigorants.

Enfin, je remercie tendrement ma famille qui m'a supportée pendant ces trois difficiles années (et les 23 précédentes). Non, je ne suis toujours pas postière malgré ce que vous en avez compris, mais grâce à vous j'ai réussi cette étape importante. Merci à mes parents, sans qui (tel à une remise d'Oscar) rien n'aurait été possible, et à mes grandes sœurs adorées Caroline et Aurélie. Merci à vous quatre pour votre soutien indéfectible et l'amour que vous me donnez. Merci aussi à Clém et Gilles, et merci à mes petits monstres de neveux Romain, Cyril, Coline, Olivia et Lexie, pour ~~le bruit~~ la joie que vous m'apportez.

Pour terminer, Burhan, je te dis encore mille fois merci pour ton soutien, ta patience, ta patience, mais également pour ta patience. Je te les rendrais au centuple quand ton tour viendra. Merci pour tout ce que tu as pu faire pour que le chemin me paraisse moins rude, comme une montagne verdoyante.

Table des matières

CHAPITRE 1

Introduction

1.1 Les systèmes distribués

Comme on peut le lire dans [Tanenbaum and van Steen, 2016],

> Un système distribué est une collection d'ordinateurs indépendants qui apparaissent aux utilisateurs du système comme un seul ordinateur. *

Cette description décrit un large spectre de systèmes et de nombreuses définitions plus précises ont été données, mais elles regroupent toutes les points énoncés dans [Ghosh, 2014] : un système distribué contient différents composants, qui sont autonomes et communiquent pour atteindre un but commun. Les systèmes distribués sont omniprésents : on les retrouve dans de nombreux contextes et à différentes échelles. En effet, le résultat est alors un tout plus efficace, avec une meilleure gestion ou permettant des actions impossibles à eux seuls. L'efficacité peut être augmentée si un calcul est divisé sur plusieurs cœurs au lieu d'un, par exemple, pour des calculs de haute performance. Un fichier partagé dans un cloud peut faciliter la gestion de données entre utilisateurs. Le web, quant à lui, constitue un exemple de comportement difficilement reproductible sans système distribué. Un autre avantage est la possibilité de partager des ressources, qui peuvent être des données comme du matériel, comme c'est le cas dans les fichiers partagés ou les réseaux d'entreprise permettant le partage d'une même imprimante.

Les langages de programmation concurrente permettent la manipulation de systèmes distribués, comme C, Go ou Erlang. S'opposant à la programmation séquentielle, on admet que plusieurs parties du code, appelées threads ou processus selon le langage, s'exécutent plus ou moins simultanément, travaillant en parallèle et parfois même en collaboration pour atteindre un but commun.

Il découle du fonctionnement d'un système distribué deux cas de gestion de mémoire : le partage de mémoire et le passage de messages. Dans le premier, la mémoire est partagée, comme un serveur accessible par plusieurs ordinateurs. Les composants sont en concurrence pour accéder aux mêmes informations, tant en lecture qu'en écriture. Il faut donc gérer les ressources partagées entre les différents éléments. Cela peut concerner des pages web consultées en même temps par des utilisateurs. Dans de tels systèmes, la réplication, qui consiste à garder en mémoire à différentes adresses la même information, peut être une solution pour améliorer la pérennité du système et réduire les temps d'accès, si les copies sont bien placées. Cependant, elle peut aussi apporter des problèmes de cohérences en cas de modification. Dans le deuxième cas, la mémoire n'est plus partagée, mais distribuée, chaque participant en possède une partie qui n'est alors pas accessible pour les autres. Pour assurer la gestion et le partage de cette mémoire, des normes telles que MPI

∗. « A distributed system is a collection of independent computers that appear to the users of the system as a single computer. », [Tanenbaum and van Steen, 2016], Chapter 1 : Introduction to Distributed Systems.

(*Message Passing Interface*) permettent de garantir que les entités aient un accès sécurisé à la donnée souhaitée, soit par envoi de participant à participant, ou encore d'un participant à tous les autres. Il peut alors permettre la gestion de deux threads dans un programme C voulant modifier la même variable.

L'intergiciel (*middleware* en anglais) permet de créer une interface entre les couches basses, contenant les protocoles de communication et la gestion de la mémoire, et les applications et services du système. Elle permet également de masquer l'hétérogénéité du système et la répartition des données. Elle prend par exemple en charge l'échange de messages entre participants.

La coordination entre ces participants indépendants est primordiale pour qu'ils atteignent leur objectif. Une synchronisation est alors nécessaire, pour cela certains systèmes, dont nous ne verrons pas les détails, possèdent des horloges globales, et d'autres utilisent les échanges de messages. Ainsi, on pourra observer des échanges de messages dans l'exemple d'un site permettant la réservation dans un hôtel. L'ordinateur du client va envoyer une requête lors de la validation de son choix. Celui du gérant enverra alors à son tour une requête au serveur où sont enregistrées les réservations, pour vérifier que la date est disponible. Il attendra ensuite une réponse avant d'approuver ou non la réservation. Ici, les messages livrent une information ou une requête. Mais ils peuvent aussi simuler une synchronisation s'ils sont utilisés comme un jeton, ils servent alors à prendre la main sur une partie de la mémoire par exemple. On voit dès lors que la communication reste inévitable et essentielle dans des contextes pourtant différents, assurant le partage d'informations et la synchronisation entre les éléments.

1.2 La vérification de systèmes distribués

Les systèmes distribués peuvent être sensibles à des pannes de différents types. Ils peuvent avoir à faire à des erreurs de conception, des accidents physiques ou des malveillances intentionnelles qui les empêchent de suivre leurs spécifications. L'ensemble de techniques de conception et d'architectures qui permettent la continuité en cas de pannes rendent possibles les tolérances aux pannes. Ceci inclut par exemple la redondance des envois des messages pour pallier une possible défaillance dans la communication. Cependant, les erreurs de conception ne peuvent pas être toujours évitées en amont. Ainsi, les systèmes distribués sont alors concernés par des problèmes spécifiques corrélés à leur fonctionnement. Les obstacles rencontrés sont bien entendu différents de ceux d'un programme séquentiel. Ils découlent principalement du fait que les processus sont indépendants et que le bon fonctionnement de l'ensemble s'appuie sur la communication.

Des erreurs peuvent alors avoir lieu, que ce soit dans le protocole élaboré à la construction ou dans le déroulement qui parfois ne peut être entièrement prévu. Ainsi, une première solution consiste à vérifier pendant le fonctionnement du système qu'aucun comportement inattendu n'a lieu, le survey [Francalanza et al., 2018] regroupe les différentes recherches sur la vérification en cours d'exécution (*runtime verification*). Mais la recherche fondamentale s'est également beaucoup intéressée à cette communication et le dernier siècle a été le témoin d'une avancée essentielle dans les théories de la concurrence dans le but de minimiser ces erreurs, notamment grâce à Milner et Hoare, auteurs de langages formels, décrivant les interactions dans un système distribué, que nous verrons plus loin. Des questions concernant la sûreté (*safety*) et d'accessibilité (*reachability*) ont été posées, l'objectif étant d'assurer que le système conçu a le comportement attendu. Plus précisément, on peut s'interroger par exemple sur l'absence d'interblocage (*deadlock* en anglais), qui est un problème de sûreté, décrivant une situation où la seule action disponible pour

chaque processus est impossible dans l'état actuel, ce qui bloque alors le système entier. Nous verrons plus en détail les autres problèmes, notamment concernant l'accessibilité, dans le Chapitre 3. Cependant, un système distribué est bien trop complexe pour être observé tel quel. Il est donc nécessaire de l'abstraire, afin d'ignorer les détails pas ou peu impactants et de préserver seulement les caractéristiques qui nous intéressent, ici, la communication.

La modélisation d'un système permet alors de se concentrer sur les informations portant sur la communication entre les processus tout en occultant les détails internes de chacun. Différents modèles existent dans la littérature.

Parmi les plus courants, on peut citer les réseaux de Petri. Ils sont représentés par des graphes où résident deux sortes de sommets : les places et les transitions. Ils constituent des éléments du système et sont respectivement désignés comme agents passifs et actifs. Les arcs relient une place à une transition. Une transition représente une action du système et permet l'ajout ou le retrait de jetons dans une place. Chaque place représentant une ressource, les jetons indiquent la disponibilité, ou la quantité disponible, de la ressource concernée. Dans un tel modèle, on pourrait simuler la présence d'un message dans un canal par un jeton dans une place, les transitions représentant alors un envoi ou une réception. Pour plus de détails sur les réseaux de Petri, voir [David and Alla, 1994].

Les algèbres de processus, qui sont des langages formels, regroupent le CCS (*Calculus of Communicating Systems*), le π-calcul ou encore le CSP (*Communicating Sequential Processes*). Le CCS, introduit par Milner dans [Milner, 1980], permet de définir des processus récursifs, mais également d'exprimer les interactions et communications entre ces derniers. Parallèlement, Hoare propose CSP dans [Hoare, 1978]. Ce langage permet lui aussi de définir les comportements des participants ainsi que le passage de messages entre eux. Le π-calcul, introduit dans [Milner, 1999] est lui aussi une algèbre de processus dérivé de CSS. Grâce à la trace de programme établi avec un tel langage, et des comparaisons telle que la bisimulation, on peut observer les comportements possibles du système et donc certifier qu'ils sont sains.

Enfin, le modèle qui retiendra notre attention sera les automates communicants [Brand and Zafiropulo, 1983]. Dans ce paradigme, chaque processus est modélisé en automate, qui, pour changer d'état, doit envoyer ou recevoir un message. Ces envois ne sont pas instantanés et les messages transitent via des canaux, où ils forment une file en FIFO (*First In First Out*). Ainsi, cette modélisation permet tout particulièrement de se focaliser sur la communication du système. C'est ce qui, associé à d'autres avantages que nous verrons par la suite, nous a poussés à nous concentrer sur ce modèle dans cette thèse. Nous nous attarderons plus longuement sur ce modèle dans le Chapitre 2.

Nous avons vu que la vérification pouvait se faire en amont comme pendant l'exécution. Les automates communicants sont appropriés à la vérification automatisée en amont grâce à des model checkers. Les model checkers sont des outils permettant le model checking, c'est-à-dire, la vérification d'un modèle pour une propriété donnée. Le plus souvent, cette propriété est exprimée avec de la logique. Chaque model checker a ses spécificités, comprenant quel type de modèle il considère. Nous citerons ci-après les plus marquants.

Pour commencer, Spin est certainement le plus connu du domaine, il permet de vérifier des systèmes distribués encodés en Promela. Promela est un langage qui autorise la modélisation d'un système en automates, ceux-ci peuvent aussi bien partager des variables globales que communiquer via des canaux. Il vise à vérifier certaines propriétés de validité, comme l'absence d'interblocage et de code inaccessibles, mais l'utilisateur peut définir ses propres propriétés logiques. De plus amples informations sur Spin pourront être trouvées dans [Holzmann and Peled, 1996].

Dans le même principe, cADP, décrit dans [Fernandez et al., 1996], analyse des systèmes dont le comportement peuvent être décrit avec du π-calcul ou d'autres langages formels, ou encore des systèmes modélisés en systèmes d'automates, communicants ou non. Cet outil permet ainsi la vérification de multiples propriétés, comme encore une fois l'absence d'interblocage, mais aussi des spécifications par rapport à une relation de bisimulations. On peut ajouter TLC, présenté par Lamport dans [Yu et al., 1999]. Il s'agit du principal outil pour la vérification de systèmes dont la spécification est faite en TLA+, autre langage formel pour modéliser des systèmes à haut niveau du même auteur. Enfin, uPPAAL, introduit dans [Bengtsson et al., 1996], permet la vérification de systèmes modélisés en automates communicants temporisés. Comme leur nom l'indique, ils permettent d'ajouter une notion de temps aux automates, par exemple un temps maximal durant lequel un automate peut rester dans un état donné avant d'en changer (plus de détails sur les automates temporisés dans [Alur and Dill, 1994]). uPPAAL supporte alors notamment la vérification de propriétés exprimées en logique temporelle.

1.3 Les automates communicants

Comme dit plus tôt, cette thèse porte sur les automates communicants. C'est un modèle populaire, introduit par [Brand and Zafiropulo, 1983], permettant de modéliser aisément systèmes distribués et protocoles de communication. Un système d'automates communicants est composé d'un ensemble d'automates représentant chacun un élément du système. Chaque transition est une action d'envoi ou de réception. Les messages transitent via des canaux, où ils sont stockés en file FIFO.

Selon la disposition des canaux, plusieurs types de communication se distinguent. Chaque machine peut avoir une boîte aux lettres où elle stocke tous les messages qu'elle reçoit, on parle alors de communication en *boîte aux lettres* (ou *mailbox* en anglais). Dans une communication en *pair à pair*, une machine aura un canal différent pour chacun des expéditeurs des messages qu'elle reçoit. Nous verrons dans de plus amples détails le fonctionnement et les subtilités de telles communications dans le Chapitre 2.

D'autres formes de communication sont aussi étudiées dans la littérature, comme dans [Clemente et al., 2014] où, en plus des précédentes, le *bag* est considéré : le stockage se fait alors dans un même ensemble sans ordre, quel que soit l'expéditeur ou le destinataire. Notre étude portera tout particulièrement sur l'ordre des messages, de ce fait nous ne considérons pas de tels types de communication.

Lorsque les canaux sont en FIFO, ils ne sont pas bornés et un nombre arbitrairement grand de messages peut alors être stocké avant d'être consommé. De ce fait, un système d'automates communicants a le pouvoir expressif d'une machine de Turing (prouvé dans [Brand and Zafiropulo, 1983]), ce qui rend la majorité des problèmes indécidables sur ce modèle. Différents subterfuges ont alors été élaborés afin de tout de même s'assurer de la validité de ces systèmes. L'idée a souvent été d'associer les comportements de ces systèmes à un ensemble de comportements finis. On peut alors étudier des systèmes aux comportements réduits, mais similaires, qui eux ne sont pas Turing-équivalents.

Pour ce faire, on peut restreindre les systèmes à une communication comparable d'une façon ou d'une autre à la communication synchrone, c'est-à-dire lorsque l'envoi et la réception des messages se font de façon simultanée. Avec une telle restriction, les propriétés telles que l'accessibilité deviennent décidables pour le système étudié. Nous regrouperons sous le nom de *synchronisabilité*

les différentes stratégies simulant cette communication synchrone. Elles seront introduites dans la suite de cette introduction puis largement explicitées dans le Chapitre 3.

1.4 La synchronisabilité

De nombreuses méthodes peuvent correspondre à cette appellation. Pour commencer, la méthode de réduction de Lipton, dans [Lipton, 1975], est décrite comme étant une analyse d'un programme parallèle en regroupant certains morceaux de code. Cette technique joue sur le fait de pouvoir interchanger des sections qui ne dépendent pas les unes des autres. En particulier, elle permet de déplacer une réception au plus près de son envoi afin de simuler la communication synchrone recherchée. C'est une idée qui sera récurrente par la suite et servira d'appui pour plusieurs travaux. À la même époque, Elrad *et al.* proposent [Elrad and Francez, 1982] où, similairement, les systèmes sont décomposés en couche, en fonction des communications, ce qui permet aux processus de se synchroniser aux frontières de ces couches. Cette décomposition simplifie alors l'analyse. Plus récemment, on note une dernière méthode qui consiste à réduire l'expressivité du système en bornant la taille des canaux. On retrouve cette technique dans [Genest et al., 2006]. Ce sont les prémices de la synchronisabilité. Les auteurs de [Chaouch-Saad et al., 2009] effectuent eux aussi une réduction, en divisant les exécutions en tours. Ici, c'est le lien entre vue globale (du système) et locale (de chacun des processus) d'une exécution qui est mis en valeur.

Plus tard, on retrouve l'idée de rapprocher les réceptions de leurs envois dans [Kragl et al., 2018], dans [v. Gleissenthall et al., 2019] ou encore dans [Basu and Bultan, 2016]. On trouve même un outil permettant de vérifier la propriété de synchronisabilité définie dans ce dernier décrit dans [Akroun and Salaün, 2018]. Enfin, les systèmes définis dans [Bouajjani et al., 2018a] combinent des idées précédentes : les exécutions admettent des phases de synchronisation, ces phases sont bornées par un nombre de messages. L'idée est encore une fois de rapprocher la réception de l'envoi, tout en admettant un peu plus de souplesse.

Bien entendu, tous les systèmes distribués ne peuvent pas se prêter à de telles analyses ou divisions, ainsi on obtient des classes de systèmes satisfaisant ces propriétés. Les systèmes synchronisables de [Basu and Bultan, 2016], existentiellement bornés de [Genest et al., 2006] et tout particulièrement k-synchronisables de [Bouajjani et al., 2018a] seront les sujets principaux de cette thèse. Ils seront définis et illustrés dans le Chapitre 3, tout comme de nouvelles variantes définies dans le cadre de cette thèse, Chapitre 6. Tous ceux-ci seront regroupés sous le nom de systèmes synchronisables.

1.5 Synopsis et contributions

Dans cette thèse, notre attention se portera donc sur les classes de systèmes synchronisables modélisés en systèmes d'automates communicants.

Le Chapitre 2 formalisera les différentes définitions, tel que les systèmes d'automates communicants et leur communication, ainsi que de différentes notions et abstractions que nous serons amenés à utiliser, telles que les diagrammes de séquences pour la représentation graphique des comportements et la notion de réalisabilité, faisant le lien entre cette représentation et le système étudié.

Le Chapitre 3 définira quant à lui les problèmes étudiés pour la vérification de systèmes ainsi que les systèmes synchronisables, permettant alors d'y répondre. Nous verrons en parti-

culier les systèmes k-synchronisables de [Bouajjani et al., 2018a], existentiellement k-bornés de [Genest et al., 2006] et synchronisables de [Basu and Bultan, 2016].

Les trois prochains chapitres présenteront les contributions apportées.

Le Chapitre 4 s'appuiera sur les travaux publiés dans [Di Giusto et al., 2020]. Ils présentent une preuve de la décidabilité de l'appartenance à la classe des systèmes k-synchronisables pour un système quelconque un k donné.

Le Chapitre 5 présente une preuve de la décidabilité de l'existence d'un k pour un système quelconque tel que le système est k-synchronisable. Il détaille les travaux publiés dans [Di Giusto et al., 2021b].

Enfin, le Chapitre 6 présentera des extensions possibles à la k-synchronisabilité, que nous nommerons k-synchronisabilité forte et la k-synchronisabilité sans ordre. Une partie des travaux présentés ont été publiés dans [Bollig et al., 2021]. Ces nouvelles classes seront alors comparées à celles déjà présentées.

Notations

Les automates

$\mathtt{A}_p$	l'automate du processus p
$L_{\mathtt{A}}$	l'ensemble des états de l'automate A
$\delta_{\mathtt{A}}$	la fonction de transition de l'automate A
ℓ_p	un état de l'automate p
$\mathcal{L}(\mathtt{A})$	le langage de l'automate A

Les ensembles et éléments d'un système

$\mathcal{S}$	un système
$\mathbb{V}$	l'ensemble des messages
$\mathbb{P}$	l'ensemble des processus
$\mathbb{A}$	l'ensemble des actions
$\mathtt{S}$	l'ensemble des envois
$\mathtt{R}$	l'ensemble des réceptions
$\mathtt{S}_p$	l'ensemble des envois du processus p
$\mathtt{R}_p$	l'ensemble des réceptions du processus p
$\overrightarrow{\ell}$	un état de contrôle global du système
γ	une configuration
$\Gamma(\mathcal{S})$	l'ensemble des configurations du système
$\Gamma_R(\mathcal{S})$	l'ensemble des configurations accessibles du système
com	le type de communication d'un système
mb	la communication en boîte aux lettres
pp	la communication en pair à pair

Les ensembles d'exécutions et de MSC

$\textsc{Ex}(\mathcal{S})$	l'ensemble des exécutions
$\textsc{Ex}^S(\mathcal{S})$	l'ensemble des exécutions synchrones
$\textsc{Ex}_k(\mathcal{S})$	l'ensemble des exécutions k-bornées
$\textsc{Msc}(\mathcal{S})$	l'ensemble des MSC
$\textsc{Msc}_k(\mathcal{S})$	l'ensemble des MSC k-synchrones
$\textsc{Msc}_k^F(\mathcal{S})$	l'ensemble des MSC k_F-synchrone du système $\mathcal{S}$
$\textsc{Msc}_k^{so}(\mathcal{S})$	l'ensemble des MSC k_{so}-synchrone du système $\mathcal{S}$

Les processus, les messages, les actions

m	un message
p, q, etc.	les processus p, q, etc.
$\mathtt{s}(p, q, m)$	l'envoi du message m par le processus p au processus q
$\mathtt{r}(p, q, m)$	la réception par le processus q du message m envoyé par le processus p
$\mathbf{m}$	le couplage du message m_i
$\mathsf{proc}_{\mathtt{S}}(\mathbf{m})$	l'expéditeur du message m
$\mathsf{proc}_{\mathtt{R}}(\mathbf{m})$	le destinataire du message m
$\mathsf{procs}(\mathbf{m})$	l'ensemble des processus concernés par le couplage $\mathbf{m}$
a	une action (d'envoi ou de réception)
e	une séquence d'actions dans $\mathbb{A}^*$
$a_i \vdash\!\dashv a_i$	les actions a_i et a_j sont couplées
$\sharp(e, a)$	le nombre d'actions a dans la séquence e

Les transitions

$\ell_p \xrightarrow{a}_p \ell'_p$	la transition dans l'automate $\mathtt{A}_p$ de l'état ℓ_p à ℓ'_p avec l'action a
$\overrightarrow{\ell} \xrightarrow{a} \overrightarrow{\ell'}$	la transition dans le système de l'état $\overrightarrow{\ell}$ à $\overrightarrow{\ell'}$ avec l'action a
$\overrightarrow{\ell} \overset{e}{\Rightarrow} \overrightarrow{\ell'}$	la transition dans le système de l'état $\overrightarrow{\ell}$ à $\overrightarrow{\ell'}$ avec la séquence d'action e
$\gamma \overset{a}{\mapsto} \gamma'$	la transition de la configuration γ à γ' avec l'action a
$\gamma \overset{e}{\Mapsto} \gamma'$	la transition de la configuration γ à γ' avec la séquence d'actions e
$\overrightarrow{\ell} \overset{\mu}{\rightsquigarrow} \overrightarrow{\ell'}$	la transition dans le système de l'état $\overrightarrow{\ell}$ à $\overrightarrow{\ell'}$ avec le MSC μ

Les canaux

$\mathbb{C}_{\mathsf{com}}(\mathbb{P}, \mathbb{V})$	l'ensemble des contenus de canaux
$\overrightarrow{c}$	le vecteur des contenus des canaux
$\overrightarrow{c_\emptyset}$	le vecteur de canaux vides
c_p	le canal du processus p en mb
$c_{p,q}$	le canal du processus q pour les messages de p en pp
$\mathbb{B}$	l'ensemble des abstractions de canaux
$\mathcal{B}(p) = (\mathcal{C}_{\mathtt{S},p}, \mathcal{C}_{\mathtt{R},p})$	l'ensemble des processus causalement liés à p
$\mathcal{B}_\pi = (\mathcal{C}^\pi_{\mathtt{S}}, \mathcal{C}^\pi_{\mathtt{R}})$	l'ensemble des processus liés à l'envoi au processus π
$\mathcal{B}^\mu(p) = (\mathcal{C}^\mu_{\mathtt{S},p}, \mathcal{C}^\mu_{\mathtt{R},p})$	l'ensemble des processus liés à p dans le MSC μ

Les MSC

μ	un MSC
Ev	l'ensemble des évènements d'un MSC
$\prec_{po}$	l'ordre partiel défini par un MSC sur les actions d'une même machine
$\prec_{src}$	l'ordre partiel défini par un MSC sur deux actions couplées
$\prec$	l'ordre partiel défini par un MSC
$\lambda(ev)$	la fonction qui associe une action à un évenement dans un MSC
$msc(e)$	le MSC de la séquence d'actions e

Les graphes de dépendances

V	l'ensemble des sommets d'un graphe
E	l'ensemble des arcs d'un graphe
$\mathsf{GD}(e)$	le graphe de dépendances de la séquence e
$\mathsf{GD}(e, \mathcal{B})$	le graphe de dépendances de la séquence e avec les sommets de synthèse déduits de $\mathcal{B}$
$\mathsf{GDE}(e)$	le graphe de dépendances étendu de la séquence e
$\xrightarrow{\text{XY}}$	l'arc XY dans un graphe de dépendances
$\overset{\text{XY}}{\dashrightarrow}$	l'arc XY dans un graphe de dépendances étendu
$\mathbf{m}_{\mathsf{start}}$	le sommet de l'action d'envoi au processus π
$\mathbf{m}_{\mathsf{stop}}$	le sommet de l'action d'envoi du processus π
ψ_p	le sommet de synthèse pour le processus p
ψ_{start}	le sommet de synthèse de $\mathbf{m}_{\mathsf{start}}$
ψ_{stop}	le sommet de synthèse de $\mathbf{m}_{\mathsf{stop}}$
$\mathsf{Post}^*(\mathbf{m})$	l'ensemble des sommets accessibles depuis $\mathbf{m}$
$\mathsf{Pre}^*(\mathbf{m})$	l'ensemble des sommets co-accessibles depuis $\mathbf{m}$
$\mathsf{Post}_e(P)$	l'ensemble des sommets accessibles depuis les processus contenus dans P dans la séquence e
$\mathsf{Pre}_e(P)$	l'ensemble des sommets co-accessibles depuis les processus contenus dans P dans la séquence e

Préliminaires

CHAPITRE 2

Modélisation et représentations

Les systèmes distribués sont complexes, car riches en informations de tous types. On peut répertorier parmi celles-ci certaines concernant les comportements individuels et les actions internes de chacun des participants. Mais on peut également y trouver des informations sur les messages, leur gestion pendant un transit entre deux processus, ou encore les conséquences de la lecture d'un d'entre eux. Les étudier nécessite des abstractions afin de ne pas devoir gérer des données, comme les actions internes, qui n'auraient pas d'impact sur la notion centrale, qui est dans cette thèse, la communication du système.

Ainsi, ce chapitre se concentre sur les différents modèles et représentations que nous utiliserons dans cette thèse. Comme dit plus tôt, nous étudierons des systèmes distribués modélisés en systèmes d'automates communicants. Dans un premier temps, nous verrons en détail la définition d'un tel modèle et sa sémantique.

La seconde partie de ce chapitre décrira des façons de représenter graphiquement les comportements du système, à travers les graphes de dépendances, permettant de visualiser les dépendances entre les actions d'une exécution et les diagrammes de séquences, représentant un ensemble d'exécutions. Nous survolerons également d'autres représentations possibles pour les exécutions présentes dans la littérature.

Enfin, ces représentations nous amèneront à la notion de relation causale entre les actions, permettant de caractériser les exécutions d'un système. Dans cette thèse, nous la nommerons réalisabilité, mais plusieurs variantes de relations causales ont été utilisées, nous les décrirons et nous pourrons les comparer à la réalisabilité.

2.1 La modélisation du système : les automates communicants

Pour modéliser un système distribué, notre choix s'est porté sur les automates communicants, parfois nommés CFM pour *Communicating Finite-state Machines*, comme par exemple dans [Kuske and Muscholl, 2010, Lange et al., 2015, Genest et al., 2007]. Ainsi, chaque machine du système sera représentée par un automate.

Rappelons qu'un automate est composé d'états et de transitions étiquetées d'une lettre. Il reconnaît un mot si celui-ci peut être écrit en suivant les transitions depuis l'état initial de l'automate. L'ensemble des mots reconnus par un automate forme son langage. Un automate est dit *fini* si le nombre d'états qui le compose est fini, et un langage est dit *régulier* s'il peut être reconnu par un automate fini. Un langage régulier peut être exprimé avec une expression régulière. Lorsque

cette description sera nécessaire, nous utiliserons les opérateurs suivants : le choix "+" et l'étoile de Kleene "*".

Les automates communicants sont alors des automates particuliers. Un système d'automates communicants est un ensemble d'automates finis qui s'échangent des messages : chaque transition est étiquetée soit par un envoi, soit par une réception.

Plus formellement, on aura alors $\mathbb{P}$ un ensemble fini de processus qui constituera le système et $\mathbb{V}$ un ensemble fini de messages commun aux processus du système. Une action d'envoi, ou simplement un envoi, est notée $\mathtt{s}(p, q, m)$ et désigne l'envoi du message m par le processus p au processus q. De la même façon, une réception $\mathtt{r}(p, q, m)$ désigne la réception par q du message m envoyé par p. Le plus souvent, on utilisera la lettre a pour parler d'une action, si l'on ne souhaite pas préciser s'il s'agit d'un envoi ou d'une réception. Les ensembles $\mathtt{S} = \{\mathtt{s}(p, q, m) \mid p, q \in \mathbb{P}, m \in \mathbb{V}\}$ et $\mathtt{R} = \{\mathtt{r}(p, q, m) \mid p, q \in \mathbb{P}, m \in \mathbb{V}\}$ désigneront respectivement l'ensemble des actions d'envoi et l'ensemble des actions de réception ; nous pourrons noter $\mathtt{X}$ ou $\mathtt{Y}$ lorsque nous parlerons d'un de ces ensembles sans savoir lequel. La notation $\mathtt{S}_p$ et $\mathtt{R}_p$ restreindra ces ensembles aux actions d'un processus $p \in \mathbb{P}$, tel que $s \in \mathtt{S}_p$ est alors de la forme $\mathtt{s}(p, q, m)$ et tel que $r \in \mathtt{R}_p$ est alors de la forme $\mathtt{r}(q, p, m)$. Chaque processus est modélisé par un automate et nous désignerons le *système* comme étant la composition parallèle de ces processus.

Un système d'automates communicants peut communiquer de différentes façons. Dans ce manuscrit, nous nous concentrerons sur deux types de communication : la communication en *boîte aux lettres* (mb) et la communication en *pair à pair* (pp). Dans le premier cas, chaque processus a son canal dans lequel il stocke tous les messages qu'il reçoit, quel que soit l'expéditeur. Dans le second, il existe un canal pour chaque paire de machines et dans chaque sens. Ainsi, il résulte que chaque machine a autant de canaux que de correspondants, et donc stocke les messages en fonction de leur expéditeur. Ces canaux seront des files en FIFO (First In First Out) dans chacune des communications. Ainsi, le premier message inséré dans la file, et donc envoyé dans le canal, sera le premier à en sortir, et donc à être lu.

Définition 2.1.1 (Système)**.** Un système est un uplet $\mathcal{S} = ((L_p, \delta_p, \ell_p^0)_{p\in\mathbb{P}}, \mathsf{com})$ où pour tout processus p, L_p est l'ensemble fini des états de contrôle locaux, $\delta_p \subseteq (L_p \times (\mathtt{S}_p \cup \mathtt{R}_p) \times L_p)$ est la fonction de transition et $\ell_p^0 \in L_p$ est l'état initial de p. La transition $(\ell, a, \ell') \in \delta_p$ pourra aussi être notée $\ell \xrightarrow{a}_p \ell'$. La variable $\mathsf{com} \in \{\mathsf{mb}, \mathsf{pp}\}$ définit le type de communication.

Dans d'autres définitions, les automates incluent des états finaux, ce n'est pas notre cas ici, ce qui est alors équivalent à avoir tous les états finaux.

Exemple 2.1.1 – Supposons un site web permettant de réserver une chambre dans différents hôtels. Un tel système est donc composé des clients connectés au site, le site lui-même et les hôtels partenaires. Simplifions les choses, et supposons qu'il n'y a qu'un client et qu'un hôtel. Nous avons alors 3 acteurs, chacun sera représenté par un automate différent dans notre système.

Pour réserver une chambre, le client doit envoyer une demande au site, contenant l'hôtel, la date et les autres options nécessaires. Sur le site, certaines dates et options sont certifiées disponibles, d'autres nécessitent la validation de l'hôtel. Ainsi, dans le premier cas, le site pourra valider directement la réservation, et prévenir l'hôtel de la réservation faite, dans le second cas, il enverra une requête à l'hôtel, celui-ci validera ou refusera la réservation. Le site web préviendra le client de la réponse, et, dans le cas d'un refus, pourra proposer une date qu'il pourra certifier disponible. Une ultime validation sera attendue dans le cas d'une nouvelle proposition.

Ainsi, avec un tel comportement, les automates modélisant ces 3 acteurs pourront être ceux représentés en Figure 2.1 dans le système nommé $\mathcal{S}_{csh}$. On remarquera que les actions internes,

qui ici peuvent englober les actions faites par des humains, telles que le refus d'une réservation par l'hôtel par exemple, sont tacites et seul le résultat, et donc le message envoyé au site web, est considéré.

De plus, on constate que le choix d'une communication en boîte aux lettres ou en pair à pair n'a, ici, pas d'importance et n'aura pas de conséquences sur le comportement du système. En effet, un seul message est stocké à la fois dans chaque canal, et le fait que le site puisse recevoir les messages du client et de l'hôtel dans le même canal ou dans deux canaux différents ne change rien.

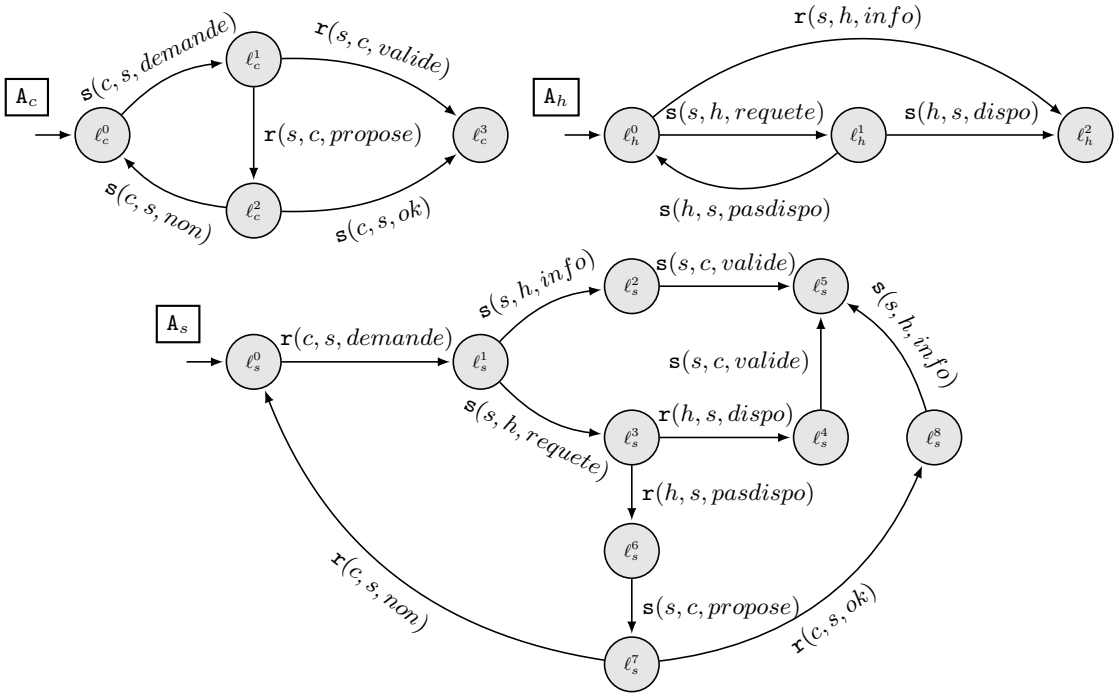


FIGURE 2.1 – Le système d'automates $\mathcal{S}_{csh}$ représentant le client ($\mathtt{A}_c$), le site web ($\mathtt{A}_s$) et l'hôtel ($\mathtt{A}_h$)

Un canal contient une suite de messages. Ainsi, le contenu d'un canal, noté c, est un mot dans $\mathbb{V}^*$. On définit le vecteur des contenus des canaux selon le type de communication.

Définition 2.1.2 (Vecteur des contenus de canaux)**.** Le vecteur des contenus des canaux noté $\vec{c}$ appartient à l'ensemble $\mathbb{C}_{\mathsf{com}}(\mathbb{P}, \mathbb{V})$ tel que

$$\mathbb{C}_{\mathsf{com}}(\mathbb{P}, \mathbb{V}) = \begin{cases} (\mathbb{V}^*)^{\mathbb{P}} & \text{si com} = \mathsf{mb} \\ (\mathbb{V}^*)^{\mathbb{P}\times\mathbb{P}} & \text{si com} = \mathsf{pp} \end{cases}$$

On notera c_p le contenu du canal du processus $p \in \mathbb{P}$ dans une communication en boîte aux lettres, et $c_{p,q}$ le contenu du canal du processus $p \in \mathbb{P}$ pour les messages envoyés par le processus $q \in \mathbb{P}$ dans une communication en pair à pair.

Nous appellerons *topologie* le graphe où les sommets représentent les processus et les arcs les directions des messages dans le système, cela permet de visualiser le sens des communications dans le système.

Définition 2.1.3 (Topologie)**.** Soit $\mathcal{S} = ((L_p, \delta_p, \ell_p^0)_{p\in\mathbb{P}}, \mathsf{com})$ un système, la topologie T de $\mathcal{S}$ est un graphe $T = (V, E)$ telle que $V = \{p \mid p \in \mathbb{P}\}$ et $E = \{p \to q \mid \exists \mathtt{s}(p, q, m) \in \mathtt{S}_p\}$

Nous pouvons relever certaines topologies particulières dont des exemples sont représentés en Figure 2.2, telle que la chaîne (*pipeline* en anglais), la topologie en anneau orienté, où le graphe crée un unique cycle, ou non orienté où le graphe est un cycle si l'on ne tient pas compte du sens des arêtes, la topologie en arbre ou encore la topologie en forêt où le système est constitué d'un ensemble d'arbres non connexes.

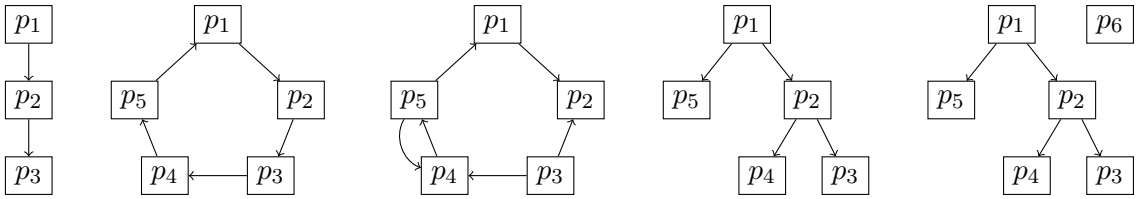


FIGURE 2.2 – Exemples de topologie : en chaîne (a), en anneau orienté (b), en anneau non orienté (c), en arbre (d), en forêt (e)

Exemple 2.1.2 – Reprenons notre système de réservation en ligne. La topologie du système est décrite par le graphe de la Figure 2.3. Elle n'appartient à aucun motif particulier vu précédemment.

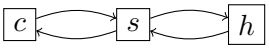


FIGURE 2.3 – La topologie du système $\mathcal{S}_{csh}$ de la Figure 2.1

Lors de l'évolution du système, on distingue l'*état de contrôle global* du système (ou simplement état de contrôle s'il n'y a pas d'ambiguïté avec l'état de contrôle d'un automate) composé des états de chacun des automates, et la *configuration* dans laquelle se trouve le système, comprenant l'état de contrôle ainsi que les contenus de chacun des canaux. Une configuration où les canaux sont vides est dite *stable*.

Définition 2.1.4 (État de contrôle global)**.** Soit $\mathcal{S} = ((L_p, \delta_p, \ell_p^0)_{p\in\mathbb{P}}, \mathsf{com})$ un système, un état de contrôle de $\mathcal{S}$ est un vecteur $\vec{\ell} = (\ell_p)_{p\in\mathbb{P}} \in \prod_{p\in\mathbb{P}} L_p$. L'état de contrôle initial que l'on notera $\vec{\ell_0}$ est défini comme $(\ell_p^0)_{p\in\mathbb{P}}$.

S'il existe une transition $\ell_p \xrightarrow{a}_p \ell'_p \in \delta_p, p \in \mathbb{P}$ alors il existe une transition $\vec{\ell} \xrightarrow{a} \vec{\ell}'$ dans le système, où l'état du processus p dans l'état de contrôle global $\vec{\ell}$, et respectivement $\vec{\ell}'$, est ℓ_p, et respectivement ℓ'_p.

Définition 2.1.5 (Configuration)**.** Soit $\mathcal{S} = ((L_p, \delta_p, \ell_p^0)_{p\in\mathbb{P}}, \mathsf{com})$ un système, une configuration de $\mathcal{S}$ est une paire $(\vec{\ell}, \vec{c})$ où $\vec{\ell}$ est un état de contrôle de $\mathcal{S}$ et $\vec{c} \in \mathbb{C}_{\mathsf{com}}(\mathbb{P}, \mathbb{V})$ est un vecteur de canaux. Le vecteur $\vec{c_\emptyset} \in \mathbb{C}_{\mathsf{com}}(\mathbb{P}, \emptyset)$ désignera le vecteur de canaux vides.

On note $\Gamma(\mathcal{S})$ l'ensemble des configurations d'un système $\mathcal{S}$.

La sémantique d'un système, permettant son évolution, est définie par les deux règles ci-dessous.

(Envoi) $$\frac{\ell_p \xrightarrow{\mathtt{s}(p,q,m)}_p \ell'_p \quad c' = c \cdot m}{(\vec{\ell}, \vec{c}) \xmapsto{\mathtt{s}(p,q,m)} (\vec{\ell'}, \vec{c'})} \qquad \text{avec } \vec{\ell'} = \vec{\ell}\ [\ell'_p/\ell_p] \text{ et } \vec{c'} = \vec{c}\ [c'/c]$$

(Réception) $$\frac{\ell_q \xrightarrow{\mathtt{r}(p,q,m)}_q \ell'_q \quad c = m \cdot c'}{(\vec{\ell}, \vec{c}) \xmapsto{\mathtt{r}(p,q,m)} (\vec{\ell'}, \vec{c'})} \qquad \text{avec } \vec{\ell'} = \vec{\ell}\ [\ell'_q/\ell_q] \text{ et } \vec{c'} = \vec{c}\ [c'/c]$$

Le canal modifié dépend du type de communication :

- si $\mathsf{com} = \mathsf{mb}$ alors $c = c_q$ et $c' = c'_q$;
- sinon, si $\mathsf{com} = \mathsf{pp}$ alors $c = c_{p,q}$ et $c' = c'_{p,q}$.

Informellement, ces règles décrivent le fait qu'une action d'envoi ajoute un message dans le canal du destinataire, tandis qu'une action de réception retire le message du dit canal.

Une exécution $e = a_1 \cdots a_n$ d'un système $\mathcal{S}$ est une séquence d'actions dans $\mathtt{S} \cup \mathtt{R}$ telle que $(\vec{\ell_0}, \vec{c_\emptyset}) \xmapsto{a_1} \cdots \xmapsto{a_n} (\vec{\ell}, \vec{c})$ pour certains $\vec{\ell}$ et $\vec{c}$. On notera alors $\overset{e}{\Rightarrow}$ pour $\xrightarrow{a_1} \cdots \xrightarrow{a_n}$ et $\overset{e}{\mapsto}$ pour $\xmapsto{a_1} \cdots \xmapsto{a_n}$. L'ensemble des exécutions du système $\mathcal{S}$ est noté $\textsc{Ex}(\mathcal{S})$.

Une exécution mb-réalisable e est une séquence d'actions telle qu'il existe un système $\mathcal{S}$ avec $\mathsf{com} = \mathsf{mb}$ où $e \in \textsc{Ex}(\mathcal{S})$. De même, une exécution pp-réalisable e est une séquence d'actions telle qu'il existe un système $\mathcal{S}$ avec cette fois $\mathsf{com} = \mathsf{pp}$ où $e \in \textsc{Ex}(\mathcal{S})$. Intuitivement, une exécution est alors réalisable si elle respecte les conditions de la communication correspondante.

Dans une séquence d'actions $e = a_1 \cdots a_n$, un envoi $a_i = \mathtt{s}(p, q, m)$ est *couplé* avec une réception $a_j = \mathtt{r}(p', q', m')$ si $i < j, p = p', q = q', m = m'$ et il existe un indice $l \geq 1$ tel que a_i et a_j sont les $l^{\grave{e}mes}$ actions de e avec ces propriétés. On notera alors $a_i \vdash a_j$. Un envoi est *non couplé* s'il n'existe pas de réception dans e qui y correspond. Par extension, on pourra parler d'un message couplé ou non couplé.

On parlera d'exécutions synchrones pour des séquences d'actions où chaque réception est faite immédiatement après son envoi. Une exécution synchrone est donc une exécution de la forme $e = e_1 \cdots e_n$ où, pour tout $1 \leq i \leq n$, e_i est soit de la forme $\mathtt{s}(p, q, m) \cdot \mathtt{r}(p, q, m)$, soit de la forme $\mathtt{s}(p, q, m)$ si m est non couplé, pour $p, q \in \mathbb{P}$ et $m \in \mathbb{V}$. L'ensemble des exécutions synchrones du système est noté $\textsc{Ex}^S(\mathcal{S}) \subseteq \textsc{Ex}(\mathcal{S})$. Notons que si e est une exécution synchrone alors elle est pp-réalisable et mb-réalisable.

On aura parfois besoin de compter le nombre d'actions correspondant à certains critères dans une séquence d'actions e. On notera alors $\sharp(e, \mathtt{s}(p, q, \_))$ le nombre d'envois fait de p à q dans e. De la même façon, $\sharp(e, \mathtt{s}(\_, q, \_)) = \sum_{p \in \mathbb{P}} \sharp(e, \mathtt{s}(p, q, \_))$ est le nombre total d'envois fait à q dans e. Par extension $\sharp(e, \mathtt{s}(\_, \_, \_)) = \sum_{q \in \mathbb{P}} \sharp(e, \mathtt{s}(\_, q, \_))$ dénombre le nombre d'actions d'envoi dans e. On définit de même les nombres de réceptions $\sharp(e, \mathtt{r}(p, q, \_))$, $\sharp(e, \mathtt{r}(\_, q, \_))$ et $\sharp(e, \mathtt{r}(\_, \_, \_))$.

Exemple 2.1.3 – Reprenons notre système de réservation en ligne. Depuis l'état initial du système $(\ell_c^0, \ell_s^0, \ell_h^0)$, la seule transition possible est celle de l'automate $\mathtt{A}_c$ étiquetée $\mathtt{s}(c, s, demande)$. Supposons une communication en boîte aux lettres, et donc $\mathsf{com} = \mathsf{mb}$, on atteint alors la configuration suivante : $((\ell_c^1, \ell_s^0, \ell_h^0), (\varepsilon, demande, \varepsilon))$. L'exécution e ci-dessous est un exemple d'exécution

du système, donc $e \in \text{EX}(\mathcal{S}_{csh})$.

$$\begin{aligned} e = &\mathtt{s}(c, s, demande) \cdot \mathtt{r}(c, s, demande) \cdot \mathtt{s}(s, h, info) \cdot \\ &\mathtt{s}(s, c, valide) \cdot \mathtt{r}(s, c, valide) \cdot \mathtt{r}(s, h, info) \end{aligned}$$

On constate que $e \notin \text{EX}^S(\mathcal{S}_{csh})$, car les envois des messages $info$ et $valide$ sont consécutifs, et l'envoi de $info$ n'est pas directement suivi de sa réception. Enfin, on constate que tous les messages de l'exécution sont couplés et que $(\ell_c^0, \ell_s^0, \ell_h^0) \overset{e}{\Rightarrow} (\ell_c^2, \ell_s^4, \ell_h^2)$.

2.2 Les représentations graphiques

Nous avons donc vu le modèle des automates communicants qui sera utilisé pour représenter le système. Cependant, celui-ci n'est pas idéal pour représenter les exécutions possibles d'un système. En effet, la sémantique des automates communicants nous permet seulement d'énumérer toutes les possibilités d'exécutions. C'est pourquoi nous utiliserons d'une part les *graphes de dépendances* afin de souligner les dépendances entre les actions d'une exécution, et d'autre part les *diagrammes de séquences* (*Message Sequence Charts* en anglais) qui nous permettront de regrouper des exécutions et de les représenter graphiquement. Nous définirons ensuite les diagrammes de séquences réalisables, pour lesquels un système existe.

2.2.1 Les graphes de dépendances

Un graphe de dépendances, aussi appelé graphe de conflits dans d'autres travaux (notamment [Bouajjani et al., 2018a]), permet de mettre en exergue les dépendances entre les différentes actions des différents messages. On peut retrouver ce modèle dans la littérature, comme dans [Bouajjani et al., 2018a], ou des modèles très proches tels que dans [Papadimitriou, 1979], [Bollig et al., 2014] ou [Charron-Bost et al., 1996]. Chaque sommet du graphe représente un message et chaque arrête entre deux sommets représente une dépendance temporelle entre deux actions des deux messages concernés. Une dépendance existe si les deux messages ont un processus en commun. Une dépendance liée à un envoi sera représentée par un S tandis qu'une dépendance liée à une réception sera représentée par un R. Plus précisément, un arc $\overset{\text{SR}}{\longrightarrow}$ depuis un message m_1 vers un autre m_2 induira une dépendance telle que la réception de m_2 ne pourra se produire qu'après l'envoi de m_1, par le même processus.

Plus formellement, on nomme *couplage* l'ensemble $\mathbf{m}$ représentant l'envoi et la réception du message m.

Définition 2.2.1 (Couplage)**.** Soit $e = a_1 \cdots a_n$ une séquence d'actions. Un couplage $\mathbf{m}$ de e est un ensemble d'indices de l'une des deux formes suivantes :

- $\mathbf{m} = \{i\}$ si a_i est un envoi non couplé,
- $\mathbf{m} = \{i, j\}$ si $a_i \vdash a_j$.

Par abus de langage, on pourra confondre $\mathbf{m}$ avec l'ensemble des actions $\{a_i, a_j\}$, ou respectivement $\{a_i\}$ si m n'est pas couplé. Soit $\mathbf{m} \ni i$ un couplage, tel que $a_i = \mathtt{s}(p, q, m)$, on note l'ensemble des processus concernés par $\mathbf{m}$ avec $\mathsf{procs}(\mathbf{m}) = \begin{cases} \{p\} & \text{si } m \text{ est non couplé} \\ \{p, q\} & \text{si } m \text{ est couplé} \end{cases}$. Enfin, que a_i soit couplée ou non, on note $\mathsf{proc}_{\mathtt{S}}(\mathbf{m})$ pour p et $\mathsf{proc}_{\mathtt{R}}(\mathbf{m})$ pour q.

Définition 2.2.2 (Graphe de dépendances)**.** Le graphe de dépendances $\mathsf{GD}(e)$ d'une séquence d'actions $e = a_1 \cdots a_n$ est un graphe étiqueté (V, E) où l'ensemble de sommets V est un ensemble de couplages tel que, pour tout $\mathtt{s}(p, q, m_i) \in e$, il existe $\mathbf{m}_i$, et pour tout X,Y $\in \{$S,R$\}$, pour tout $\mathbf{m}, \mathbf{m}' \in V$, il existe un arc de dépendance XY $\mathbf{m} \xrightarrow{\text{XY}} \mathbf{m}' \in E$ entre $\mathbf{m}$ et $\mathbf{m}'$ s'il existe $i < j$ tels que $i \in \mathbf{m}, j \in \mathbf{m}', a_i \in \mathtt{X}, a_j \in \mathtt{Y}$, et $\mathsf{proc}_{\mathtt{X}}(\mathbf{m}) = \mathsf{proc}_{\mathtt{Y}}(\mathbf{m}')$.

On notera $\mathbf{m} \to \mathbf{m}'$ s'il existe $\mathbf{m} \xrightarrow{\text{XY}} \mathbf{m}' \in E$, et ainsi, on notera $\mathbf{m} \to^* \mathbf{m}'$ s'il existe un chemin (possiblement vide) de $\mathbf{m}$ à $\mathbf{m}'$. Parfois, si deux actions ne sont pas consécutives sur une machine, on pourra s'abstenir de dessiner l'arc correspondant sur la figure si celui-ci n'a pas d'impact.

Exemple 2.2.1 – Soit e l'exécution du système $\mathcal{S}_{csh}$ déjà vu dans l'Exemple 2.1.3 :

$$e = \mathtt{s}(c, s, demande) \cdot \mathtt{r}(c, s, demande) \cdot \mathtt{s}(s, h, info) \cdot \\ \mathtt{s}(s, c, valide) \cdot \mathtt{r}(s, c, valide) \cdot \mathtt{r}(s, h, info)$$

Son graphe de dépendances est représenté par la Figure 2.4, où chaque sommet représente des d'actions couplées tel que :

$$\begin{aligned} \mathbf{m}_d &= \{\mathtt{s}(c, s, demande), \mathtt{r}(c, s, demande)\}, \\ \mathbf{m}_i &= \{\mathtt{s}(s, h, info), \mathtt{r}(s, h, info)\}, \\ \text{et } \mathbf{m}_v &= \{\mathtt{s}(s, c, valide), \mathtt{r}(s, c, valide)\}. \end{aligned}$$

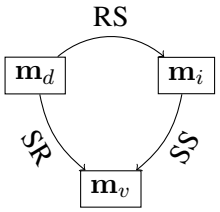


FIGURE 2.4 – Le graphe de dépendances de l'exécution $e \in \text{EX}(\mathcal{S}_{csh})$

2.2.2 Les diagrammes de séquences

Une exécution impose un ordre total sur les actions qu'elle contient. Mais celui-ci n'est pas systématiquement nécessaire, et l'on va voir que l'on peut intervertir certaines actions sans aucun impact sur le comportement global du système. Ainsi se fait sentir une nécessité d'abstraire les exécutions afin de ne pas conserver des informations inutiles telles que peut l'être l'ordre total. Nous venons donc à regrouper les exécutions sous une représentation préservant seulement un ordre partiel sur les actions. Nous introduisons les diagrammes de séquences (Message Sequence Chart en anglais, d'où l'abréviation MSC que nous utiliserons) qui gardent seulement l'ordre au sein des actions couplées et entre les actions d'un même processus.

Un diagramme de séquences permet de décrire les actions séquentielles de machines travaillant en parallèle. Bien qu'utilisé auparavant, la standardisation de ce modèle fut publiée dans

[CCITT, 1992, Grabowski et al., 1993], puis complétée et enrichie, ajoutant par exemple la possibilité de combiner plusieurs diagrammes de séquences en introduisant la notion de HMSC, jusqu'à sa dernière version à ce jour, publiée en 2011 [Recommendation, 2011]. On trouve aussi un formalisme équivalent dans le langage UML également appelé diagrammes de séquences.

Les diagrammes de séquences peuvent servir d'outil pour la représentation d'un comportement, comme c'est le cas dans [Charron-Bost et al., 1996], ils peuvent permettre l'analyse du système via des propriétés qui les caractérisent comme le font les auteurs de [Alur et al., 1996] mais également l'analyse logique du système, comme cela est fait dans [Wolfgang et al., 1997] ou [Bollig et al., 2018].

Informellement, un diagramme de séquences est représenté par des lignes verticales, une pour chaque processus, où le temps défile de haut en bas. Des évènements sont représentés par des points sur les lignes et correspondent à des actions, envois ou réceptions. Une flèche relie deux actions couplées, de l'envoi à la réception. Si un envoi n'est pas couplé, la flèche est en pointillés et nous permet de visualiser tout de même le destinataire. Enfin, un diagramme de séquences admet un ordre partiel, composé de l'ordre total des évènements sur chaque machine, l'ordre total entre deux actions d'un même message ainsi que leur fermeture transitive.

Définition 2.2.3 (Diagramme de séquences)**.** Un diagramme de séquence (ou MSC pour *Message Sequence Chart*) μ est un uplet $\mu = (Ev, \lambda, \prec_{po}, \prec_{src})$ où

- Ev est un ensemble fini d'évènements,
- $\lambda : Ev \to \mathtt{S} \cup \mathtt{R}$ associe une action à chaque évènement,
- $\prec_{po}$ est un ordre partiel sur Ev tel que, pour tout processus p, $\prec_{po}$ induit un ordre total sur l'ensemble des évènements de p, c'est-à-dire, sur $\lambda^{-1}(\mathtt{S}_p \cup \mathtt{R}_p)$,
- $\prec_{src}$ est une relation binaire qui relie chaque évènement de réception à son évènement d'envoi correspondant :
  - pour tout évènement $r \in \lambda^{-1}(\mathtt{R})$, il y a exactement un évènement s tel que $s \prec_{src} r$,
  - pour tout évènement $s \in \lambda^{-1}(\mathtt{S})$, il y a au plus un évènement r tel que $s \prec_{src} r$,
  - pour tous évènements s, r tels que $s \prec_{src} r$, il y a p, q, m tel que $\lambda(s) = \mathtt{s}(p, q, m)$ et $\lambda(r) = \mathtt{r}(p, q, m)$.
- $\prec = (\prec_{po} \cup \prec_{src})^+$, la fermeture transitive de $\prec_{po}$ et $\prec_{src}$, est irréflexive.

Dans la suite de la thèse, nous utiliserons le terme MSC. Dans [Lamport, 1978], Lamport définit un ordre partiel sur les actions dans une exécution, appelée *happened before*, que l'on peut traduire par "se produire avant". Cet ordre partiel est identique à l'ordre $\prec$ défini par le MSC d'une exécution. Notre définition autorise l'absence d'une réception pour un envoi et donc les messages non couplés, ce qui ajoutera des subtilités par la suite.

Pour toute séquence d'actions *bien-formée* (où toute réception est couplée) $e = a_1 \cdots a_n$, on définit $msc(e)$ le MSC où

- $Ev = [1..n]$,
- $\prec_{po}$ est composé des paires d'indices (i, j) tels que $i < j$ et $\{a_i, a_j\} \subseteq \mathtt{S}_p \cup \mathtt{R}_p$ pour un certain $p \in \mathbb{P}$, autrement dit, a_i et a_j sont des actions du même processus,

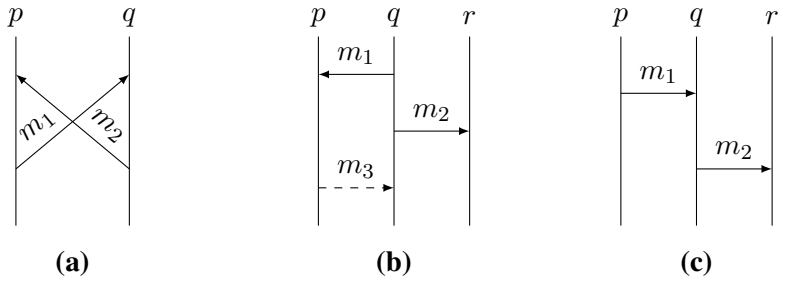
✢ $\prec_{src}$ est composé des paires d'indices (i, j) telles que $a_i \vdash a_j$.

On dit que $e = a_1 \cdots a_n$ est une *linéarisation* de μ si $\mu = msc(e)$. Une linéarisation de μ est alors un ordre total sur l'ordre partiel $\prec$ de μ. On parlera d'*exécutions causalement équivalentes* pour deux exécutions d'un système e, e' qui ont le même MSC, c'est-à-dire, $msc(e) = msc(e')$.

Exemple 2.2.2 – Pour cet exemple, nous nous appuierons sur les MSC de la Figure 2.5.

✢ La Figure 2.5.a ne représente pas un MSC. En effet, l'ordre entre les actions est réflexif, par exemple :

$$\lambda^{-1}(\mathtt{s}(p,q,m_1)) \prec \lambda^{-1}(\mathtt{r}(p,q,m_1)) \prec \lambda^{-1}(\mathtt{s}(q,p,m_2)) \prec$$
$$\lambda^{-1}(\mathtt{r}(q,p,m_2)) \prec \lambda^{-1}(\mathtt{s}(p,q,m_1))$$



(a) **(b)** **(c)**

FIGURE 2.5 – Un diagramme qui n'est pas un MSC (a), un MSC quelconque (b) et $msc(e_3)$ de l'Exemple 2.2.2 (c)

✢ Examinons maintenant un MSC. La Figure 2.5.b décrit une exécution contenant trois messages, m_1, m_2 et m_3, et trois processus, p, q et r. Le message m_1 est envoyé par q et reçu par p. q est aussi l'expéditeur du message m_2, son destinataire est la machine r. Enfin, le message m_3 est envoyé par le processus p au processus q mais ne sera jamais reçu, il n'est pas couplé.

Si l'on regarde machine par machine, on observe que le processus p doit lire le message m_1 avant d'envoyer le message m_3. De la même façon, le processus q doit envoyer m_1 pour pouvoir envoyer m_2. On a alors $\lambda(\mathtt{s}(q,p,m_1)) \prec \lambda(\mathtt{s}(q,r,m_2))$.

Plusieurs exécutions peuvent correspondre à ce MSC, comme e_1 et e_2, qui sont donc des linéarisations :

$$e_1 = \mathtt{s}(q,p,m_1) \cdot \mathtt{r}(q,p,m_1) \cdot \mathtt{s}(q,r,m_2) \cdot \mathtt{r}(q,r,m_2) \cdot \mathtt{s}(p,q,m_3)$$
$$e_2 = \mathtt{s}(q,p,m_1) \cdot \mathtt{s}(q,r,m_2) \cdot \mathtt{r}(q,p,m_1) \cdot \mathtt{s}(p,q,m_3) \cdot \mathtt{r}(q,r,m_2)$$

Ici, on voit que le MSC permet une exécution synchrone, l'exécution e_1, ce qui n'est pas toujours le cas.

✢ Enfin, soit $e_3 = \mathtt{s}(p,q,m_1) \cdot \mathtt{r}(p,q,m_1) \cdot \mathtt{s}(q,r,m_2) \cdot \mathtt{r}(q,r,m_2)$ une exécution mb-réalisable. On note $<$ l'ordre total donné par l'exécution e_3. On peut prendre l'ordre des actions de e_3 sur chacun des processus ainsi qu'entre des actions couplées :

- ✻ sur la machine p, il n'y a que l'action $\mathtt{s}(p, q, m_1)$,
- ✻ sur la machine q :

$$\mathtt{r}(p, q, m_1) < \mathtt{s}(q, r, m_2) \text{ donc } \lambda^{-1}(\mathtt{r}(p, q, m_1)) \prec_{po} \lambda^{-1}(\mathtt{s}(q, r, m_2))$$

- ✻ sur la machine r, il n'y a que l'action $\mathtt{r}(q, r, m_2)$,
- ✻ entre les actions couplées :

$$\mathtt{s}(p, q, m_1) \mapsto \mathtt{r}(p, q, m_1) \text{ donc } \lambda^{-1}(\mathtt{s}(p, q, m_1)) \prec_{src} \lambda^{-1}(\mathtt{r}(p, q, m_1))$$
$$\text{et } \mathtt{s}(q, p, m_2) \mapsto \mathtt{r}(q, p, m_2) \text{ donc } \lambda^{-1}(\mathtt{s}(q, p, m_2)) \prec_{src} \lambda^{-1}(\mathtt{r}(q, p, m_2))$$

On peut alors en déduire le MSC de la Figure 2.5.c qui représente donc $msc(e_3)$.

On définit l'ensemble des MSC du système $\mathcal{S}$ comme $\text{MSC}(\mathcal{S}) = \{msc(e) \mid e \in \text{EX}(\mathcal{S})\}$. On pourra noter $\vec{\ell} \overset{\mu}{\rightsquigarrow} \vec{\ell}'$ si $\vec{\ell} \overset{e}{\Rightarrow} \vec{\ell}'$ avec e une linéarisation de μ. On peut relever que si $\vec{\ell} \overset{e}{\Rightarrow} \vec{\ell}'$ pour e une linéarisation de μ, alors $\vec{\ell} \overset{e'}{\Rightarrow} \vec{\ell}'$ pour tout autre linéarisation e' de μ. Notez que, pour une séquence d'action e quelconque, le fait que $msc(e) \in \text{MSC}(\mathcal{S})$ n'implique pas que $e \in \text{EX}(\mathcal{S})$, autrement dit, toute linéarisation d'un MSC n'est pas systématiquement une exécution.

Exemple 2.2.3 – Voyons plusieurs exemples de linéarisations.

- ✢ Le MSC de la Figure 2.6.a ne représente aucune linéarisation qui soit une exécution pp-réalisable (et donc non plus mb-réalisable). En effet, la seule linéarisation possible est :

$$e_1 = \mathtt{s}(p, q, m_1) \cdot \mathtt{s}(p, q, m_2) \cdot \mathtt{r}(p, q, m_2) \cdot \mathtt{r}(p, q, m_2)$$

Celle-ci n'est pas réalisable dans un système communiquant en pair à pair, le message m_1 étant envoyé avant dans le canal $c_{p,q}$ en premier et occupant donc la tête de la file, le message m_2 ne peut être lu en premier. C'est également le cas en boîte aux lettres, une linéarisation ne pouvant être non pp-réalisable mais mb-réalisable.

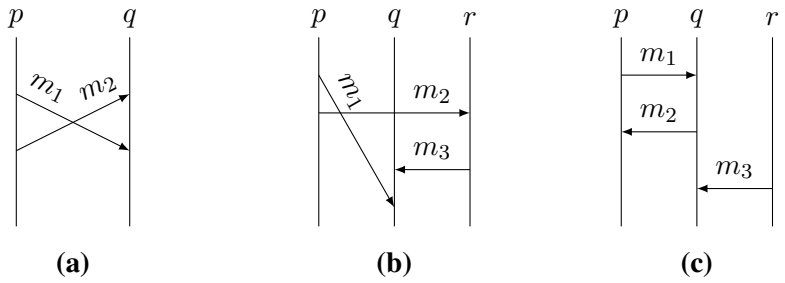


FIGURE 2.6 – Des MSC pour l'Exemple 2.2.3

- ✢ Pour le MSC de Figure 2.6.b, une seule linéarisation est possible :

$$e_2 = \mathtt{s}(p, q, m_1) \cdot \mathtt{s}(p, r, m_2) \cdot \mathtt{r}(p, r, m_2) \cdot \mathtt{s}(r, q, m_3) \cdot \mathtt{r}(r, q, m_3) \cdot \mathtt{r}(p, q, m_1)$$

e_2 est pp-réalisable, m_1 est stocké dans le canal $c_{p,q}$ pendant le transfert des autres messages jusqu'à la réception du message m_3 qui permet la réception du m_1. Elle n'est cependant pas mb-réalisable, puisque dans une communication en boîte aux lettres, les messages m_1 et m_3 seraient envoyés tous deux dans le canal c_q, avec m_1 en tête, ce qui empêcherait la lecture de m_3 de se faire avant.

- Enfin, pour un même MSC, on peut avoir des linéarisations correspondant certaines à des exécutions pp-réalisables et mb-réalisables, et d'autres à des exécutions seulement pp-réalisables, comme c'est le cas avec la Figure 2.6.c. Pour ce MSC, on a deux linéarisations intéressantes :

$$e_3 = \mathtt{s}(p,q,m_1)\cdot\mathtt{r}(p,q,m_1)\cdot\mathtt{s}(q,p,m_2)\cdot\mathtt{r}(q,p,m_2)\cdot\mathtt{s}(r,q,m_3)\cdot\mathtt{r}(r,q,m_3)$$
$$e_4 = \mathtt{s}(r,q,m_3)\cdot\mathtt{s}(p,q,m_1)\cdot\mathtt{r}(p,q,m_1)\cdot\mathtt{s}(q,p,m_2)\cdot\mathtt{r}(q,p,m_2)\cdot\mathtt{r}(r,q,m_3)$$

  e_3 est une exécution synchrone, et donc pp-réalisable et mb-réalisable. Cependant, e_4 n'est pas mb-réalisable, en effet, si l'envoi de m_3 se fait en premier dans l'exécution, la réception de m_1 ne pourra être la première action de q.

On définit également comme suit la concaténation de deux MSC μ_1, μ_2 concernant le même ensemble de processus. Intuitivement, cela consiste à mettre le second sous le premier.

Définition 2.2.4 (Concaténation de MSC)**.** Soient deux MSC $\mu_1 = (Ev_1, \lambda_1, \prec^1_{po}, \prec^1_{src})$ et $\mu_2 = (Ev_2, \lambda_2, \prec^2_{po}, \prec^2_{src})$ tels que $Ev_1 \cap Ev_2 = \emptyset$.
La concaténation $\mu_1 \cdot \mu_2$ est le MSC $\mu = (Ev, \lambda, \prec_{po}, \prec_{src})$ tel que :

- $Ev = Ev_1 \cup Ev_2$
- $\lambda = \lambda_1 \cup \lambda_2$
- $\prec_{po} = \prec^1_{po} \cup \prec^2_{po} \cup \bigcup_{p\in\mathbb{P}}\{(e_1,e_2) \mid e_1 \in \lambda_1^{-1}(\mathtt{S}_p \cup \mathtt{R}_p), e_2 \in \lambda_2^{-1}(\mathtt{S}_p \cup \mathtt{R}_p)\}$
- $\prec_{src} = \prec^1_{src} \cup \prec^2_{src}$.

Exemple 2.2.4 – Pour ce dernier exemple sur les MSC, nous nous appuierons sur la Figure 2.7.

- Commençons avec la Figure 2.7.a. Les messages m_1 et m_2 sont ici entrelacés. Le processus p doit envoyer m_1 pour recevoir m_2 et le processus q doit envoyer m_2 pour recevoir m_1. Un tel MSC ne peut avoir de linéarisation qui soit une exécution synchrone. $e_1 = \mathtt{s}(p,q,m_1)\cdot\mathtt{s}(q,p,m_2)\cdot\mathtt{r}(p,q,m_1)\cdot\mathtt{r}(q,p,m_2)$ est un exemple de linéarisation possible, il s'agit d'une exécution pp-réalisable et mb-réalisable.

- Reprenons notre site de réservation et son système $\mathcal{S}_{csh}$, la Figure 2.7.b représente $msc(e)$, $e \in \mathrm{Ex}(\mathcal{S}_{csh})$ de l'Exemple 2.1.3. Rappelons que $e \notin \mathrm{Ex}^S(\mathcal{S})$ mais il existe une linéarisation $e' \in \mathrm{Ex}^S(\mathcal{S})$ telle que $msc(e') = msc(e)$.

MSC et graphes de dépendances

On peut remarquer que toutes les linéarisations d'un MSC ont le même graphe de dépendances. On peut donc parler d'un MSC et du graphe de dépendances associé. On pourra donc par extension définir le graphe de dépendances d'un MSC μ, noté $\mathsf{GD}(\mu)$: $\mathsf{GD}(\mu) = \mathsf{GD}(e)$ telle que $\mu = msc(e)$.

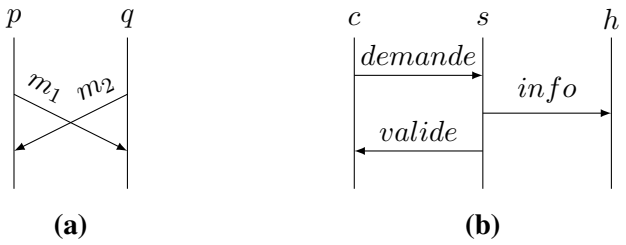


FIGURE 2.7 – Quelques MSC

Exemple 2.2.5 – La Figure 2.8.b est le graphe de dépendances associé au MSC de la Figure 2.8.a. On constate que l'envoi de m_1 est nécessaire à la réception de m_2 et inversement, il en résulte un cycle dans le graphe de dépendances. De plus, le message m_3 a besoin de l'envoi de m_2 ainsi que la réception de m_1 pour être envoyé, on retrouve donc les deux arcs depuis $\mathbf{m}_1$ et $\mathbf{m}_2$ vers $\mathbf{m}_3$.

La Figure 2.8.d est le graphe de dépendances du MSC de la Figure 2.8.c, tel qu'il existe une exécution $e \in \text{EX}(\mathcal{S}_{csh})$ et $msc(e)$ est le MSC représenté en Figure 2.8.c. $\mathbf{m}_e$, $\mathbf{m}_r$, $\mathbf{m}_i$ et $\mathbf{m}_v$ représentent respectivement les envois et réceptions des messages étiquetés $demande, requete, dispo$ et $valide$. On retrouve alors que la réception du message $demande$ est nécessaire à l'envoi du message $requete$ avec l'arc RS, que l'envoi du message $dispo$ doit avoir lieu, pour ensuite être reçu et permettre l'envoi de $dispo$, que $dispo$ doit être reçu pour que l'envoi de $valide$ ait lieu, et enfin que l'envoi du message $demande$ doit avoir lieu pour que le message $valide$ puisse être reçu. Ici, graphiquement, on a ignoré certains liens, comme l'arc $\xrightarrow{\text{RS}}$ qui existe entre $\mathbf{m}_e$ et $\mathbf{m}_v$.

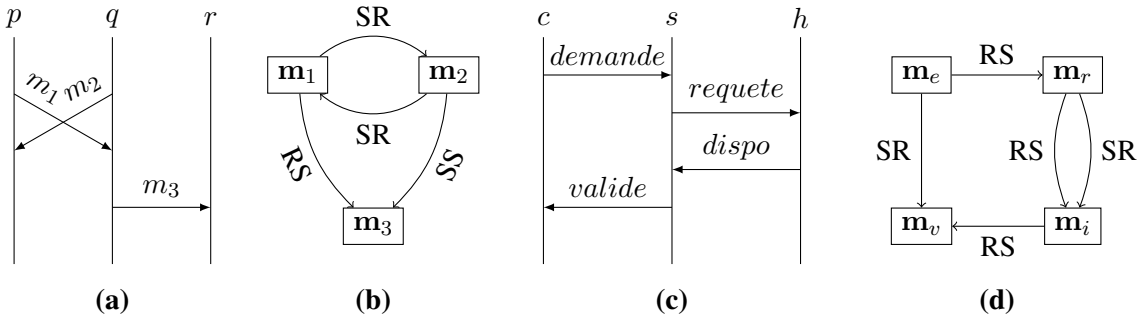


FIGURE 2.8 – Des MSC accompagnés de leurs graphes de dépendances

Des extensions aux MSC

Comme les présente l'étude [Harel and Thiagarajan, 2003], des extensions aux MSC existent dans le but d'élargir leur expressivité. On voit par exemple les diagrammes de séquences en direct (*Live Sequence Charts*) proposés dans [Damm and Harel, 2001]. Il s'agit d'une version multimodale des diagrammes de séquences que nous utilisons, qui permet notamment de distinguer des évènements possibles, nécessaires ou interdits. Pour finir, on peut aussi relever les diagrammes

de séquences haut niveau (*High Level Message Sequence Chart, HMSC*, parfois appelés *Message Sequence Graph, MSG*), définis tout comme les MSC dans [CCITT, 1992] qui permettent la spécification d'un système grâce à une sorte d'automate où les états sont des MSC, différenciant un état initial et des états finaux.

2.2.3 La réalisabilité

Comme nous l'avons vu, toute linéarisation d'un MSC ne correspond pas toujours à une exécution réalisable dans un type de communication donné. Notre objectif est donc ici de déterminer si l'ordre partiel d'un MSC admet au moins une exécution réalisable dans la communication voulue. Plus précisément, nous voulons savoir s'il existe une linéarisation du MSC qui soit une exécution dans un système communiquant via des canaux en FIFO disposés soit en boîte aux lettres, soit en pair à pair. Si c'est le cas, on pourra dire que le MSC étudié est réalisable. Nous commencerons par étudier le cas de la communication en boîte aux lettres, et verrons dans un second temps la communication en pair à pair.

La réalisabilité dans la communication en boîte aux lettres

Comme nous l'avons vu, le but est de définir une propriété exprimant la réalisabilité d'un MSC avec des canaux FIFO disposés en boîte aux lettres : l'ordre partiel induit par le MSC doit admettre un ordre total cohérent avec une telle communication, deux messages entrant dans un certain ordre dans un canal doivent en ressortir dans le même ordre. Dans le cas d'un message non couplé qui resterait stocké dans un canal, les messages envoyés plus tard dans ce même canal doivent eux aussi être non couplés, car ils ne peuvent être lus.

Nous obtenons donc la définition suivante :

Définition 2.2.5 (Réalisabilité en boîte aux lettres)**.** Soit $\mu = (Ev, \lambda, \prec_{po}, \prec_{src})$ un MSC. On dit alors que μ est mb-*réalisable* s'il existe une linéarisation $e = a_1 \cdots a_n$ avec un ordre total $<$ telle que, pour toute paire d'évènements $i < j$ telle que $a_i = \mathtt{s}(p, q, m)$ et $a_j = \mathtt{s}(p', q, m')$, soit a_j est non couplé, soit il existe i', j' tel que $a_i \vdash\!\dashv a_{i'}$, $a_j \vdash\!\dashv a_{j'}$ et $i' < j'$.

S'il n'y a pas d'ambiguïté sur le type de communication dans le contexte, nous pourrons simplement parler d'un MSC *réalisable*.

Exemple 2.2.6 – Le MSC de la Figure 2.9.a représente un ensemble d'exécutions mb-réalisables et est donc lui aussi mb-réalisable. Il est facile de voir qu'une exécution telle que l'exécution e suivante est cohérente dans un système communiquant en boîte aux lettres.

$$e = \mathtt{s}(p, q, m_1) \cdot \mathtt{s}(p, r, m_2) \cdot \mathtt{r}(p, r, m_2) \cdot \mathtt{s}(r, q, m_3) \cdot \mathtt{r}(p, q, m_1) \cdot \mathtt{r}(r, q, m_3)$$

Le MSC de la Figure 2.9.b représente lui aussi une exécution mb-réalisable. En effet, l'on peut envoyer les messages m_2 et m_3 avant d'envoyer le m_1, auquel cas le fait que ce dernier soit non couplé ne pose aucun problème. Cependant, toutes les linéarisations de ce MSC ne sont pas mb-réalisables : si l'envoi de m_1 se fait avant l'envoi de m_3, la lecture de m_3 ne pourra se faire. Cependant, l'existence d'une linéarisation correspondant à une exécution mb-réalisable est suffisante pour dire que ce MSC est mb-réalisable.

Avec une telle définition, nous obtenons la propriété visée suivante :

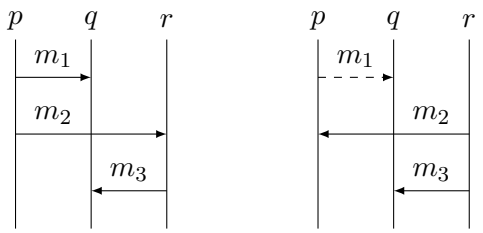


FIGURE 2.9 – Des MSC mb-réalisables

Propriété 1. *Un MSC μ est* mb*-réalisable si et seulement s'il existe un système $\mathcal{S}$ avec* com $=$ mb *et une exécution $e \in \text{Ex}(\mathcal{S})$ tels que $\mu = msc(e)$.*

Remarque 2.2.1 – Le terme de réalisabilité est également utilisé dans [Alur et al., 2003]. On doit cependant distinguer la définition de celle énoncée ici. Dans [Alur et al., 2003], elle s'applique à un ensemble de MSC qui est alors réalisable s'il existe un système tel que cet ensemble est compris dans l'ensemble de MSC du système. La réalisabilité est également définie dans un sens plus large dans [Kuske and Muscholl, 2010], où alors une spécification quelconque correspond à un système (pour plus de détails, voir [Kuske and Muscholl, 2010, Partie 3]).

La réalisabilité dans la communication en pair à pair

Dans les systèmes communiquant en pair à pair, chaque canal est attribué non plus seulement à un destinataire, mais à une paire expéditeur - destinataire, les contraintes sont alors relâchées. Un message non couplé envoyé par un processus p à un processus q n'empêche alors plus un processus r d'envoyer un message couplé à q, on le constate dans le MSC de la Figure 2.10.a. Un autre cas particulier ressort, on est à présent capable de lire des messages dans un ordre différent de celui de leur envoi, à condition qu'ils aient des expéditeurs différents. On le retrouve dans le diagramme de séquences de la Figure 2.10.b.

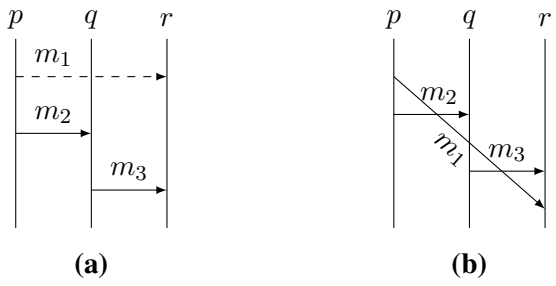


FIGURE 2.10 – Des MSC pp-réalisables

Exemple 2.2.7 – Le MSC de la Figure 2.10.a représente un ensemble d'exécutions possibles dans un système communiquant en pair à pair, ce qui n'est pas le cas en boîte aux lettres. Plus précisé-

ment, l'ordre partiel du MSC implique que le message m_1 doit être envoyé en premier. Cependant, dans une communication en boîte aux lettres, il sera alors stocké dans le canal c_r, occupant la tête de la file et empêchant le message m_3 d'être reçu. Dans une communication en pair à pair, le message m_1 sera stocké dans le canal $c_{p,r}$, et le message m_3 transmis via le canal $c_{q,r}$ peut alors être lu.

Le MSC de la Figure 2.10.b représente une exécution pp-réalisable. En effet, le message m_1 doit être reçu en dernier, mais envoyé en premier. Cela ne pose ici pas de problème puisqu'il sera stocké dans le canal $c_{p,r}$ en attendant de pouvoir être lu. Dans une communication en boîte aux lettres, ce MSC ne serait pas réalisable.

Enfin, la seule contrainte que l'on se doit de respecter reste l'ordre dans un même canal, donc l'ordre de deux messages avec les mêmes expéditeur et destinataire. Ainsi, les MSC des Figures 2.11.a et 2.11.b ne correspondent à aucune exécution et ne sont pas pp-réalisables.

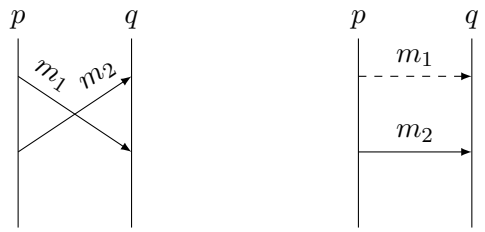


FIGURE 2.11 – Des MSC non pp-réalisables

Définition 2.2.6 (Réalisabilité en pair à pair)**.** Soit $\mu = (Ev, \lambda, \prec_{po}, \prec_{src})$ un MSC. On dit alors que μ est pp-*réalisable* s'il existe une linéarisation $e = a_1 \cdots a_n$ avec un ordre total $<$ telle que, pour toute paire d'évènements $i < j$ telle que $a_i = \mathtt{s}(p, q, m)$ et $a_j = \mathtt{s}(p, q, m')$, soit a_j est non couplé, soit il existe i', j' tel que $a_i \vdash a_{i'}$, $a_j \vdash a_{j'}$ et $i' < j'$.

On notera que la seule différence avec la définition précédente réside dans le fait que les expéditeurs de m et m' sont les mêmes. Comme précédemment, nous pourrons qualifier un MSC de *réalisable*, au lieu de pp-réalisable, si le contexte n'induit aucune ambiguïté.

Similairement à la communication en boîte aux lettres, on observe la propriété suivante.

Propriété 2. *Un MSC μ est* pp*-réalisable si et seulement s'il existe un système $\mathcal{S}$ avec* com $=$ pp *et une exécution $e \in \mathrm{Ex}(\mathcal{S})$ tels que $\mu = msc(e)$.*

2.3 La relation causale dans la littérature et comparaison

Notre notion de réalisabilité est inspirée de la notion de livraison causale (*causal delivery* en anglais) telle que dans [Bouajjani et al., 2018a]. La livraison causale est elle-même inspirée d'autres propriétés sur les MSC. Dans cette section, nous verrons alors les différentes propriétés existantes, et terminerons par une comparaison, tout particulièrement entre la livraison causale et la réalisabilité.

2.3.1 Des relations causales

On réunit sous le terme de relation causale les différentes propriétés qui relient des actions dans un MSC. Dans la littérature, différentes propriétés correspondent à ce terme, *causal ordering* ou *causal delivery*. Elles s'appuient souvent sur l'ordre *happened before* évoqué plus tôt. On retrouve dans ces propriétés l'idée qu'une exécution doit correspondre au type de communication du système et ne pas aller à l'encontre de ce que le système est capable de faire. Il faut respecter la causalité entre les messages.

L'ordre *happened before*

L'ordre partiel *happened before* est défini dans [Lamport, 1978]. L'auteur expose le problème suivant : dans un système distribué, l'ordre entre les actions effectuées par le système est parfois indéterminable. Cependant, certaines informations sont, elles, évidentes, dues à la construction du système lui-même. On peut les résumer par l'ordre partiel *happened before*, identique à l'ordre partiel $\prec$ définit par un MSC. Un évènement "se produit avant" un autre si l'on est capable de faire un chemin dans le MSC en considérant qu'on ne peut emprunter la flèche d'un message que dans le sens de la flèche, et qu'on ne peut emprunter la ligne d'un processus qu'en respectant le sens du temps, du haut vers le bas.

L'idée de causalité est aussi introduite. La relation *happened before* de [Lamport, 1978] est souvent appelée *relation causale* dans d'autres travaux, comme [Charron-Bost et al., 1996].

Définition 2.3.1 (Évènements causalement liés, évènements concurrents)**.** Soit μ un MSC défini tel que $\mu = (Ev, \lambda, \prec_{po}, \prec_{src})$. Si deux évènements $e_1 \prec e_2 \in Ev$ alors e_1 et e_2 sont causalement liés. De plus, deux évènements e_1 et e_2 sont dits *concurrents* si $e_1 \not\prec e_2$ et $e_2 \not\prec e_1$.

La figure qui suit nous permet de visualiser cet ordre sur des MSC. Chaque flèche relie deux évènements tels que le premier doit "se produire avant" le second. Nous utiliserons cette représentation pour l'ordre partiel $\prec$ dans les prochains exemples. On constate en particulier que dans le MSC de la Figure 2.12.a, les envois de m_1 et m_2 sont causalement liés, ce qui n'est pas le cas dans le MSC de la Figure 2.12.b et dans celui de la Figure 2.12.c, où ce sont alors des évènements concurrents.

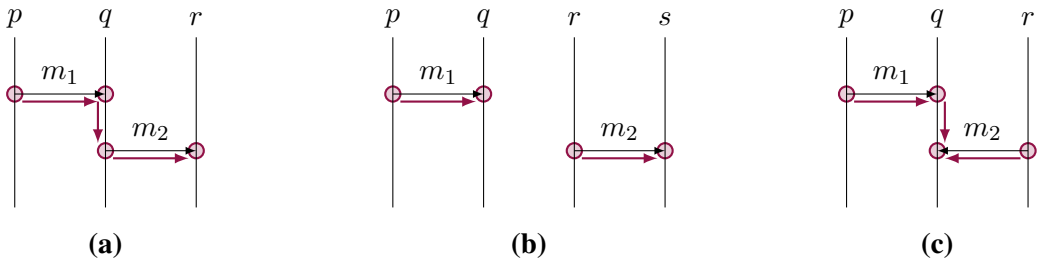


FIGURE 2.12 – Des représentations de la relation $\prec_{\mathsf{hb}}$

Cet ordre partiel fut depuis son introduction la base d'autres travaux. C'est le cas dans [Birman and Joseph, 1987], où les auteurs définissent un protocole de communication forçant à respecter un ordre partiel constitué en partie de l'ordre *happened before* adapté, avec des variantes, à la communication en *broadcast* de leurs systèmes. C'est également le cas pour l'article

[Charron-Bost et al., 1996] qui utilise cet ordre partiel pour trouver un ordre total arbitraire sur les évènements du système tel qu'il permet, si l'on considère un système distribué avec partages de ressources par exemple, d'empêcher les cas de conflits dans la mémoire. Dans ces travaux, les auteurs de [Charron-Bost et al., 1996] ont effectué un travail considérable, s'appliquant à définir différentes relations sur les actions qu'ils ont également hiérarchisées. Ils considèrent aussi bien les exécutions synchrones que les exécutions asynchrones, où les premières impliquent qu'un envoi est bloquant, la prochaine action est toujours la réception du message envoyé, et où les secondes permettent d'autres actions entre l'envoi et la réception d'un même message. Nous nous appuierons entre autres sur ces travaux, ainsi que sur ceux de [Chevrou et al., 2016], pour explorer l'état de l'art.

Les exécutions FIFO

Parmi les exécutions spécifiées dans [Charron-Bost et al., 1996], on distingue par exemple les *exécutions FIFO* qui correspondent à ce que nous appelons exécutions pp-réalisables dans cette thèse, à la différence qu'elles ne considèrent pas les messages non couplés. Elles correspondent également aux exécutions notées $1-1$ dans d'autres notations, notamment [Chevrou et al., 2016], et [Basu and Bultan, 2016]. La propriété est définie comme suit.

Définition 2.3.2 (Exécution FIFO [Charron-Bost et al., 1996])**.** Soit $\mu = (Ev, \lambda, \prec_{po}, \prec_{src})$ un MSC, μ admet une exécution FIFO si, pour deux messages $m, m' \in \mathbb{V}$, tels que $\mathbf{m} = \{s, r\}$ et $\mathbf{m}' = \{s', r'\}$:

$$(\mathsf{proc}_{\mathtt{S}}(\mathbf{m}) = \mathsf{proc}_{\mathtt{S}}(\mathbf{m}')) \wedge (\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \mathsf{proc}_{\mathtt{R}}(\mathbf{m}')) \wedge (s \prec s') \implies r \prec r' \tag{2.1}$$

Les exécutions causalement ordonnées

On distingue aussi les exécutions appelées *causally ordered*, soit *causalement ordonnées* en français.

Définition 2.3.3 (Exécution causalement ordonnée [Charron-Bost et al., 1996])**.** Soit un MSC $\mu = (Ev, \lambda, \prec_{po}, \prec_{src})$, μ admet une exécution causalement ordonnée si, pour deux messages $m, m' \in \mathbb{V}$, tels que $\mathbf{m} = \{s, r\}$ et $\mathbf{m}' = \{s', r'\}$:

$$(\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \mathsf{proc}_{\mathtt{R}}(\mathbf{m}')) \wedge (s \prec s') \implies r \prec r' \tag{2.2}$$

Intuitivement, si une exécution est causalement ordonnée, cela assure qu'un message entre deux processus ne peut être dépassé par une chaîne de message faite en parallèle. Concrètement, cela évite le cas que l'on peut voir dans le MSC de la Figure 2.13 où le message m_3 est envoyé d'abord, mais est reçu après les messages m_1 et m_2. Ce MSC n'admet alors aucune exécution causalement ordonnée. Si une exécution n'est pas causalement ordonnée, alors aucune exécution équivalente, c'est-à-dire, avec le même MSC, ne pourra correspondre à une exécution synchrone.

On constate alors qu'une exécution vérifiant la Formule 2.2 vérifie toujours la Formule 2.1. Ainsi, une exécution causalement ordonnée est FIFO. La réciproque n'est pas vraie.

Exemple 2.3.1 – Regardons des exemples d'exécutions FIFO.

- Prenons pour exemple le MSC de la Figure 2.13. On a là un MSC qui ne correspond à aucune exécution causalement ordonnée. On constate que $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}_2) = \mathsf{proc}_{\mathtt{R}}(\mathbf{m}_3) = r$ et

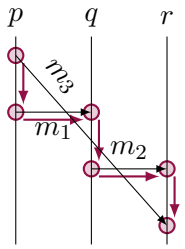


FIGURE 2.13 – Le MSC d'une exécution FIFO mais ne correspondant à aucune exécution causalement ordonnée

que, comme le montre le chemin de causalité sur la figure, $\mathtt{s}(p, r, m_3) \prec \mathtt{s}(q, r, m_2)$. Alors, pour être causalement ordonnée, une exécution doit admettre $\mathtt{r}(p, r, m_3) < \mathtt{r}(q, r, m_2)$. Or $\mathtt{r}(q, r, m_2) \prec \mathtt{r}(p, r, m_3)$, et donc aucune exécution représentée par ce MSC ne peut être causalement ordonnée. Cependant, ce MSC représente bien des exécutions FIFO.

✢ Observons un MSC permettant une exécution causalement ordonnée. Le MSC de la Figure 2.14 permet en effet de valider la Formule 2.2 : on voit que $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}_1) = \mathsf{proc}_{\mathtt{R}}(\mathbf{m}_2) = q$ mais ni $\mathtt{s}(p, q, m_1) \not\prec \mathtt{s}(r, q, m_2)$, ni l'inverse. On peut le constater en suivant les chemins de causalité représentés sur le MSC : les envois de m_3 et de m_2 sont concurrents. Ainsi, la formule logique est vraie, la propriété est respectée et l'on peut trouver une exécution causalement ordonnée.

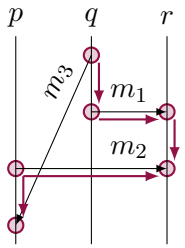


FIGURE 2.14 – Le MSC d'une exécution causalement ordonnée

De multiples caractérisations pour les exécutions causalement ordonnées sont également définies et étudiées dans le reste de l'article [Charron-Bost et al., 1996].

Les exécutions RSC

Nous pouvons aussi nous interroger sur l'existence d'une linéarisation synchrone, c'est-à-dire, si pour un MSC, il existe une linéarisation qui est une exécution synchrone, où chaque réception est immédiatement subséquente à l'envoi correspondant. C'est le but de la propriété RSC (*Réalisable avec une Communication Synchrone*).

Définition 2.3.4 (MSC RSC [Charron-Bost et al., 1996])**.** Soit $\mu = (Ev, \lambda, \prec_{po}, \prec_{src})$ un MSC, μ est RSC s'il existe une linéarisation avec un ordre total $<$ tel que $\nexists a \in \mathbb{A}, a_i < a < a_j$ pour chaque couplage $\mathbf{m} = \{i, j\}$.

Par extension, une exécution est RSC si son MSC est RSC.

Exemple 2.3.2 – Le MSC de la Figure 2.15 représente un MSC RSC, tandis que les MSC des Figures 2.14 et 2.13 ne permettent pas de linéarisations correspondant à des exécutions synchrones.

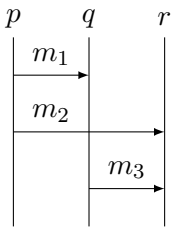


FIGURE 2.15 – Un MSC vérifiant la propriété RSC

Une hiérarchie est établie par [Charron-Bost et al., 1996] entre les différentes exécutions que nous venons voir. On peut la visualiser dans la Figure 2.16. On constate que : les exécutions synchrones sont des exécutions RSC, les exécutions RSC sont causalement ordonnées et les exécutions causalement ordonnées sont FIFO.

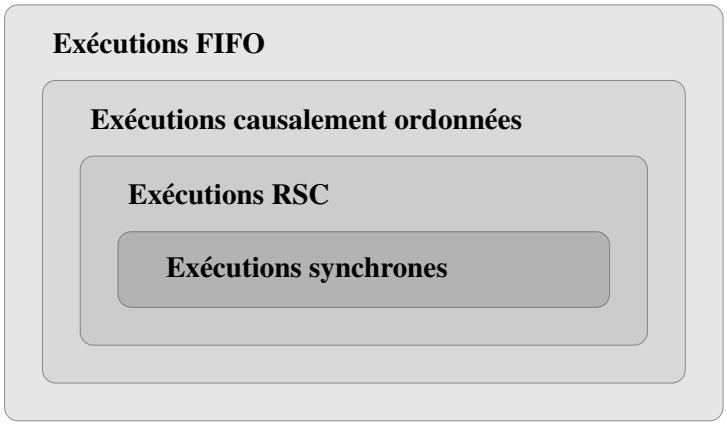


FIGURE 2.16 – Hiérarchie des exécutions définies dans [Charron-Bost et al., 1996]

Des implémentations de systèmes ont parfois été conçues tout spécialement pour s'assurer de respecter de telles propriétés. Dans [Mattern and Fünfrocken, 1995], les auteurs décrivent une implémentation telle qu'un système ne puisse produire que des exécutions causalement ordonnées. L'objectif est d'empêcher un message entre deux processus d'être dépassé par une chaîne de processus faite en parallèle. Une telle contrainte sur les systèmes étudiés, qui communiquent en pair à pair via des canaux en FIFO, implique alors d'être RSC. Intuitivement, ces systèmes sont constitués de file FIFO d'envois et de réceptions pour chaque processus. Chaque message envoyé est alors mis instantanément en queue de cette file. Il sera transmis au canal du destinataire uniquement après que la queue d'envoi aura reçu un accusé de réception du précédent message envoyé. Cette technique assure que deux messages envoyés par le même processus à un même destinataire ne pourront être lus dans un ordre différent de celui de l'envoi, tout en ne forçant pas les envois à être des actions bloquantes.

Les ordres renforcé et déduit

Dans [Alur et al., 1996], les auteurs définissent une analyse pour les MSC. L'ordre partiel d'un MSC est appelé dans ces travaux ordre visuel, mais diffère de celui défini plus tôt. On y distingue un ordre *renforcé*, sous-entendu par le type de communication, et d'un ordre *déduit*, sous-entendu de la transitivité de l'ordre renforcé. La différence réside donc dans la prise en compte du type communication du système. Une contrainte est alors introduite : deux envois concurrents à un même destinataire dans un système communiquant en boîte aux lettres doivent être envoyés selon un ordre dépendant de l'ordre des réceptions, afin de respecter les canaux en FIFO. Ainsi, dans la Figure 2.17, le message m_1 doit être envoyé avant le message m_2 pour être une exécution valide en boîte aux lettres. Ceci rejoint notre définition d'une exécution mb-réalisable. Plusieurs types de communication sont étudiés de la sorte, y compris la communication en pair à pair et des communications avec des canaux en *bag*, où les messages sont stockés sans ordre.

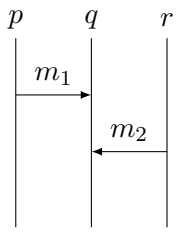


FIGURE 2.17 – Un MSC pour illustrer la contrainte des communications boîte aux lettres

Relation de préséance

La notion de relation causale entre messages a été enrichie au fur et à mesure. Ainsi a-t-elle évoluée, tout comme dans [Muscholl et al., 1998], [Muscholl and Peled, 1999]. Globalement, ces articles ont respectivement pour but de définir des propriétés et problèmes décidables sur les MSC et les HMSC. Leurs études s'appuient sur une variante de la relation causale. L'ordre partiel $\prec$ peut, d'après les auteurs, être ambigu et impliquer un ordre qui n'est propre qu'à la représentation visuelle et non pas à la construction du système. Ainsi, les auteurs introduisent une nouvelle relation, basée sur cet ordre $\prec$, nommée *relation de préséance*. Le but est de ne pas forcément conserver l'ordre total sur les actions d'une même machine si celui-ci peut dépendre de la sémantique du système. Pour cela, il utilise un autre outil de modélisation que sont les traces de Mazurkiewicz (voir [Kuske and Muscholl, 2010] pour plus de détails).

La communication $n-1$

De par les nombreuses définitions présentes dans la littérature, la nécessité de clarifier les ambiguïtés qu'elles impliquent s'est fait sentir. C'est ce que les auteurs de [Chevrou et al., 2016] se sont appliqués à faire en comparant différents types de communication et ordres partiels. Ils présentent alors, parmi les ordres vus dans cette section, la relation de causalité définie par Lamport, les exécutions FIFO comme présentées plus tôt ainsi que le fait d'être réalisable en synchrone (RSC). Nous pouvons noter une différence, les exécutions étudiées dans [Chevrou et al., 2016] sont considérées comme des ordres totaux sur les actions, à la différence de la plupart des travaux

précédents, travaillant majoritairement sur l'ordre partiel des MSC. Ainsi, une exécution est RSC si chaque envoi est suivi de sa réception, ce qui revient à être synchrone, et non pas s'il existe un ordre total sur l'ordre $\prec$ comme dit précédemment.

Ils décrivent notamment la communication $n-1$ en FIFO qui correspond à la communication en boîte aux lettres utilisée dans cette thèse. Cependant, il réside quelques différences entre les contraintes que nous imposons et les exécutions $n-1$ ainsi définies, dues à l'absence de messages sans réception que nous considérons dans nos travaux, et à l'ordre total $<$ de l'exécution qui est pris en compte par les auteurs.

Définition 2.3.5 (Exécution $n-1$)**.** Pour une exécution avec un ordre total $<$ sur ses actions, pour deux messages $m, m' \in \mathbb{V}$, tels que $\mathbf{m} = \{s, r\}$ et $\mathbf{m'} = \{s', r'\}$:

$$(\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \mathsf{proc}_{\mathtt{R}}(\mathbf{m'})) \wedge (s < s') \implies r < r' \tag{2.3}$$

On constate une forte ressemblance avec les exécutions causalement ordonnées, elles aussi considérées dans [Chevrou et al., 2016]. Cependant, elles ne sont pas identiques, et l'on peut trouver des cas qui ne satisfont pas les deux propriétés à la fois. La définition d'une exécution $n-1$ est cependant semblable à la définition d'une exécution mb-réalisable, à la différence que les messages non couplés ne sont pas considérés. Cependant, la communication $n-1$ n'est vue que par le prisme des exécutions et non par celui des MSC, ce qui fait qu'il n'y a pas d'équivalent aux MSC mb-réalisables.

Exemple 2.3.3 – La Figure 2.18 nous présente un MSC qui représente des exécutions causalement ordonnées, mais ne représente aucune exécution $n-1$. Plus précisément, si on considère toute paire de messages envoyés au même destinataire, il nous faut regarder les paires m_1-m_2, et m_3-m_4.

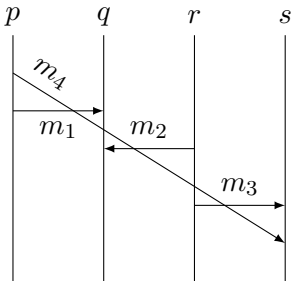


FIGURE 2.18 – Un MSC représentant des exécutions causalement ordonnées, mais ne représentant pas d'exécution $n-1$

Commençons par la Formule 2.2. Les envois dans chacune des paires sont concurrents, c'est-à-dire : $\mathtt{s}(p, q, m_1) \not\prec \mathtt{s}(r, q, m_2)$, et $\mathtt{s}(r, s, m_3) \not\prec \mathtt{s}(p, s, m_4)$. Ainsi, les exécutions représentées par ce MSC sont bien causalement ordonnées. Cependant, si l'on regarde maintenant la Formule 2.3, on a $\mathtt{r}(p, q, m_1) \prec \mathtt{r}(r, q, m_2)$ donc, il nous faut $\mathtt{s}(p, q, m_1) < \mathtt{s}(r, q, m_2)$. Comme $\mathtt{s}(p, s, m_4) \prec \mathtt{s}(p, q, m_1)$ et $\mathtt{s}(r, q, m_2) \prec \mathtt{s}(r, s, m_3)$, alors $\mathtt{s}(p, s, m_4) < \mathtt{s}(r, s, m_3)$. Cependant, $\mathtt{r}(r, s, m_3) \prec \mathtt{r}(p, s, m_4)$ donc $\mathtt{r}(p, s, m_4) \not\prec \mathtt{r}(r, s, m_3)$. Autrement dit, nous ne sommes pas capables de trouver une linéarisation de $\prec$ qui soit une exécution mb-réalisable, et le MSC de la Figure 2.18 n'est pas mb-réalisable.

D'autres types de communication sont étudiés dans [Chevrou et al., 2016], tel qu'une communication $n - n$ en FIFO, où tous les messages sont stockés dans la même file quels que soient les expéditeurs et destinataires, ou une communication parfaitement asynchrone, correspondant donc à la communication *bag* vue dans [Alur et al., 1996]. En plus de comparer et hiérarchiser ces types de communication, une méthode de vérification de la communication asynchrone y est également décrite.

2.3.2 La livraison causale et la réalisabilité

Enfin, nous définirons ici la livraison causale utilisée dans l'article [Bouajjani et al., 2018a] puis nous la comparerons avec la réalisabilité, définie plus haut, Définition 2.2.5.

La livraison causale

Les travaux dans [Bouajjani et al., 2018a] utilisent la notion de *causal delivery*, que l'on pourrait traduire par *livraison causale*.

Définition 2.3.6 (Livraison causale [Bouajjani et al., 2018a])**.** Soit $\mu = (Ev, \lambda, \prec_{po}, \prec_{src})$ un MSC, μ vérifie la livraison causale si, pour deux messages $m, m' \in \mathbb{V}$, tels que $\mathbf{m} = \{s, r\}$ et $s' \in \mathbf{m}'$:

$$(s \prec s') \wedge (\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \mathsf{proc}_{\mathtt{R}}(\mathbf{m}')) \implies (\mathbf{m}' = \{s'\}) \vee (\mathbf{m}' = \{s', r'\} \wedge (r' \not\prec r)) \quad (2.4)$$

Si l'on décortique cette formule, on constate en premier lieu que l'on considère des messages qui peuvent ne pas avoir de réceptions et donc être non couplés. Plus précisément, on voit que si deux messages sont envoyés dans un certain ordre à un même processus, quels que soient les expéditeurs, soit le second doit être non couplé, soit les réceptions doivent être faites dans un ordre correspondant à celui des envois. L'objectif est donc d'être causalement ordonné (comme vu notamment dans [Charron-Bost et al., 1996]) tout en prenant en compte les messages non couplés.

On peut remarquer entre les exécutions vérifiant la livraison causale et les exécutions $n-1$ définies dans [Chevrou et al., 2016] la même différence que nous avons relevée entre les exécutions causalement ordonnées de [Charron-Bost et al., 1996] et les exécutions $n - 1$ (cf. Exemple 2.3.3 Figure 2.18). On ne tient ici pas compte des messages non couplés, n'étant pas considérés par les autres propriétés.

Exemple 2.3.4 – La Figure 2.19.a représente un MSC avec des messages non couplés vérifiant la livraison causale, et ce parce que les envois des messages m_1 et m_2 sont concurrents. En effet, si l'on regarde la Formule 2.4, un ordre est imposé sur les réceptions uniquement si les envois sont causalement liés. La formule est alors vérifiée et le MSC vérifie la livraison causale.

Figure 2.19.b représente lui aussi un MSC vérifiant la livraison causale, cette fois-ci les envois des messages m_1 et m_2 sont causalement liés, mais la réception de m_2 est absente, ainsi le second envoi est non couplé et la Formule 2.4 est validée.

Comparaison entre livraison causale et réalisabilité

Pour cette section, nous ne considérerons que la réalisabilité dans une communication en boîte aux lettres dans le but de la comparer à la livraison causale.

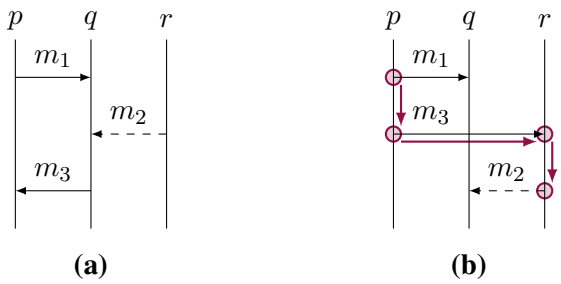


FIGURE 2.19 – Des MSC vérifiant la livraison causale

La livraison causale comme la réalisabilité considère des canaux en FIFO et des messages non couplés. Cependant, un MSC peut satisfaire la livraison causale sans correspondre à une exécution mb-réalisable. C'est pourquoi nous avons entrepris d'introduire la notion de réalisabilité, plus forte, et permettant de certifier la cohérence avec la communication en boîte aux lettres.

En effet, la gestion des messages non couplés est incomplète pour assurer cette dernière. Dans un premier temps, pour qu'une exécution soit mb-réalisable, si deux messages sont envoyés avec un lien causal au même destinataire, soit le second message est non couplé, soit les réceptions doivent se faire dans l'ordre attendu. La Formule 2.4 se contente de vérifier que les réceptions n'aient pas à se faire dans l'ordre inverse de celui des envois, d'après l'ordre partiel du MSC.

Exemple 2.3.5 – Le MSC de la Figure 2.20.a vérifie la livraison causale, mais n'atteint pas notre objectif de représenter une exécution mb-réalisable. En effet, le message m_1 doit être envoyé avant le message m_3, comme le confirme le chemin de causalité entre les actions. Sauf que, le message m_1 est non couplé et restera donc dans le canal de la machine r et empêchera les messages envoyés plus tard dans ce canal d'être lus. Le message m_3 qui doit être couplé ne pourra donc pas être lu. D'un autre côté, la Formule 2.4 est vérifiée : les réceptions n'ont pas d'ordre particulier, la première n'existant pas.

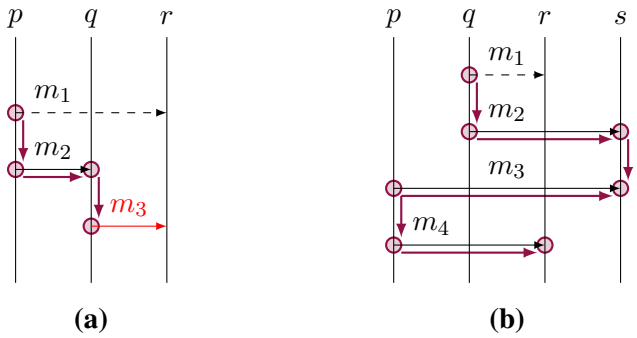


FIGURE 2.20 – Des MSC problématiques

Dans un second temps, si une exécution est mb-réalisable, deux messages envoyés à la même machine et reçus dans un certain ordre impliquent le même ordre sur les envois de ces messages. Ce n'est pas le cas pour vérifier la livraison causale.

Exemple 2.3.6 – Le chemin causal que nous pouvons suivre dans le diagramme de la Figure 2.20.b indique que les envois des messages m_1 et m_4 sont concurrents. On en conclut donc que le MSC vérifie la livraison causale. Cependant, on constate que la communication en boîte aux lettres n'est ici pas prise en compte et en effet, nous avons $\mathtt{r}(q, s, m_2) \prec \mathtt{r}(p, s, m_3)$ et la communication en FIFO et en boîte aux lettres impose que le message m_2 doive alors être envoyé avant le m_3 pour leur permettre d'être lus dans cet ordre. Ainsi, toute exécution valable dans cette sémantique, où l'ordre entre les actions est définie par $<$, aura $\mathtt{s}(q, s, m_2) < \mathtt{s}(p, s, m_3)$. Comme $\mathtt{s}(q, r, m_1) \prec \mathtt{s}(q, s, m_2)$ et $\mathtt{s}(p, s, m_3) \prec \mathtt{s}(p, r, m_4)$, on attend comme contrainte que $\mathtt{s}(q, r, m_1) < \mathtt{s}(p, r, m_4)$. On constate alors que si l'on force cette contrainte, le MSC n'est alors plus compatible avec la propriété précédente qui assure qu'aucun message couplé ne soit envoyé après un message non couplé à un même processus, ici les messages m_1, non couplé, et m_4, couplé.

Pour terminer cette comparaison, regardons un dernier exemple en détail.

Exemple 2.3.7 – Le MSC de la Figure 2.21 vérifie la livraison causale, mais n'est pas mb-réalisable. En effet, il n'existe comme chemin de causalité que les chemins de m_1 à m_3 et de m_2 à m_4. Ainsi, les seuls envois causaux ne concernent pas les mêmes destinataires et la propriété est donc vérifiée.

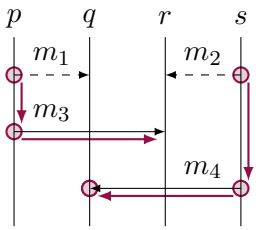


FIGURE 2.21 – Un MSC non mb-réalisable vérifiant la livraison causale

Cependant, essayons de créer une exécution à partir de ce MSC :

- Imaginons que le message m_1 soit le premier à être envoyé, alors le canal de la machine q est condamné pour le reste de l'exécution, ce qui contredit la réception de m_4 qui est couplé ;
- Prenons alors m_2 comme premier message envoyé, de la même façon que précédemment c'est le canal de r qui restera condamné, on ne pourra donc pas avoir m_3 couplé comme indiqué ;
- Il paraît évident que ni l'envoi du message m_3 ni celui de m_4 ne peuvent être la première action de l'exécution, puisqu'ils nécessitent respectivement l'envoi de m_1 et l'envoi de m_2.

Ainsi, aucune linéarisation ne pourrait convenir aux comportements des canaux FIFO en boîte aux lettres. Effectivement, supposons que la linéarisation établit que :

- $\mathtt{s}(p, q, m_1) < \mathtt{s}(s, q, m_4)$ alors on cherche à avoir m_4 non couplé, ce qui n'est pas le cas quelle que soit la linéarisation, soit $\mathtt{r}(p, q, m_1) < \mathtt{r}(s, q, m_4)$ ce qui n'est pas le cas non plus, puisque l'action $\mathtt{r}(p, q, m_1)$ ne peut apparaître dans la linéarisation ;

- ✢ $\mathtt{s}(s,q,m_4) < \mathtt{s}(p,q,m_1)$ alors, comme $\mathtt{s}(s,r,m_2) \prec_{po} \mathtt{s}(s,q,m_4)$, nous avons $\mathtt{s}(s,r,m_2) < \mathtt{s}(p,q,m_1)$. Symétriquement, $\mathtt{s}(p,q,m_1) \prec_{po} \mathtt{s}(p,r,m_3)$ donc également $\mathtt{s}(s,r,m_2) < \mathtt{s}(p,r,m_3)$. Finalement, on est donc dans un cas similaire, où m_2, non couplé, est envoyé avant m_3, couplé, au processus r, ce qui ne satisfera pas les contraintes.

Aucune linéarisation ne peut donc satisfaire les contraintes, donc ce MSC n'est pas mb-réalisable.

2.4 Conclusion

Ce chapitre aura présenté les notions nécessaires aux développements qui suivront dans les prochains chapitres. Nous avons notamment décrit le modèle des automates communicants et les représentations graphiques que nous utiliserons : les MSC et les graphes de dépendances.

Nous avons également détaillé les différentes relations causales entre les actions qui peuvent caractériser les exécutions. Bien que nous utiliserons la notion de réalisabilité dans cette thèse, nous avons pu la comparer à des variantes proches comme la livraison causale.

Le prochain chapitre permettra alors d'introduire les problèmes de vérification sur ce modèle que sont les automates communicants, ainsi que différentes heuristiques pour y répondre.

CHAPITRE 3

La vérification du modèle

Modéliser un sujet d'étude permet d'y vérifier l'absence d'erreurs. Nous avons alors affaire à différents problèmes de vérification que nous verrons dans ce chapitre. Des réponses positives à ces problèmes garantient la confiance que nous pouvons avoir dans le système. Nous modélisons un système distribué à l'aide d'automates communicants, définis dans le chapitre précédent. Ce choix a ses avantages mais aussi des inconvénients. En effet, certains problèmes de vérification y sont indécidables.

Parmi les différents problèmes de vérification, nous nous intéressons au problème de l'accessibilité. Malheureusement, l'ensemble des configurations accessibles n'est pas calculable pour des systèmes quelconques. On a donc recours au calcul soit d'une sur-approximation, soit d'une sous-approximation de l'ensemble des configurations accessibles. Elles consistent dans le système en la restriction de certains paramètres, ou bien en la relaxation de certaines contraintes. Une méthode de calcul approché peut s'avérer exacte pour une certaine classe de systèmes : cette classe de systèmes définit alors une sous-classe de systèmes communicants pour laquelle le problème de l'accessibilité est décidable.

Nous étudierons ces différentes stratégies avec un intérêt particulier pour deux sous-classes qui reviendront dans les chapitres à venir.

Ce chapitre se décompose alors comme ceci : nous verrons dans un premier temps les problèmes pertinents dans notre étude, puis deux sous-classes de systèmes communicants que sont les systèmes k-synchronisables et k-bornés. Nous terminerons par un tour d'horizon des approches par sous et sur-approximations présentes dans la littérature.

3.1 Des problèmes de vérification

Nous commençons donc par définir les problèmes les plus courants : les problèmes d'accessibilité. Pour cela, il nous faut définir la notion de configuration accessible.

Définition 3.1.1 (Configuration accessible)**.** Soit $\mathcal{S} = ((L_p, \delta_p, \ell_p^0)_{p\in\mathbb{P}}, \mathsf{com})$ un système. Une configuration $\gamma \in \Gamma(\mathcal{S})$ est accessible s'il existe $e \in \textsc{Ex}(\mathcal{S})$ telle que $(\vec{\ell_0}, \vec{c_\emptyset}) \overset{e}{\Longmapsto} \gamma$.

On note $\Gamma_R(\mathcal{S}) = \{\gamma \mid \gamma \in \Gamma(\mathcal{S}) \text{ est accessible}\}$ l'ensemble des configurations accessibles. On peut alors définir les problèmes d'accessibilité.

Définition 3.1.2 (Accessibilité d'une configuration)**.** Le problème de l'accessibilité d'une configuration est le problème de décision suivant :

Entrée : un système $\mathcal{S}$, une configuration $\gamma \in \Gamma(\mathcal{S})$

Question : γ appartient-elle à l'ensemble des configurations accessibles $\Gamma_R(\mathcal{S})$?

Définition 3.1.3 (Accessibilité d'un état de contrôle)**.** Le problème de l'accessibilité d'un état de contrôle est le problème de décision suivant :

Entrée : un système $\mathcal{S}$, un état de contrôle $\vec{\ell} \in L_{\mathcal{S}}$

Question : Existe-t-il $\vec{c} \in \mathbb{C}_{\mathsf{com}}(\mathbb{P}, \mathbb{V})$ et $\gamma = (\vec{\ell}, \vec{c}) \in \Gamma(\mathcal{S})$ telle que $\gamma \in \Gamma_R(\mathcal{S})$?

Le problème de l'accessibilité est le plus simple, il est d'autant plus intéressant que, parfois, d'autres problèmes peuvent être réduits à un problème d'accessibilité.

On trouve différents types de problèmes. Certains problèmes sont dits *de sûreté* : il s'agit de s'assurer que, pour un système et un ensemble, possiblement infini, de configurations problématiques, aucune de ces configurations n'est accessible. Lorsque l'ensemble de ces configurations est fini, cela revient effectivement à un problème d'accessibilité.

Dans cette thèse, nous nous interesserons uniquement aux problèmes d'accessibilité. Nous allons toutefois mentionner ci-dessous d'autres problèmes de sûreté. D'une part, on parlera de situations d'interblocage, qui décrivent une situation où tout les processus sont dans des états où seules des transitions de réception sont disponibles, mais que toutes les files sont vides. Les processus s'attendent mutuellement et le système est alors bloqué. Un système est alors dit sans interblocages si aucune configuration correspondant à une situation d'interblocage n'appartient à l'ensemble des configurations accessibles. Si un processus n'a que des transitions de réception de disponibles, mais que le message en tête de file ne correspond à aucune transition, il s'agit alors d'une situation de message inattendu (*unexpected message*). La recherche d'une configuration avec une telle situation correspond également à un problème de sûreté.

Bien d'autres types de problèmes existent. [Sistla, 1994] en a proposé une classification dans laquelle figurent les problèmes dits *de vivacité*. Ils consistent à s'assurer qu'une propriété est vérifiée au bout d'un certain temps fini, et ce quel que soit l'ordonnancement.

Si les automates communicants donnent une interprétation graphique et intuitive pour étudier des systèmes distribués, notamment communiquant en FIFO, ils ont le principal inconvénient d'être Turing-équivalents, comme prouvé dans [Brand and Zafiropulo, 1983], ce qui signifie d'une part que l'on peut simuler une machine de Turing avec de tels systèmes, mais d'autre part que les problèmes indécidables sur les machines Turing le sont également pour nos systèmes. Ainsi, les problèmes d'accessibilité sont indécidables.

De nombreuses heuristiques pour résoudre le problème de l'accessibilité ont été proposées. On peut en citer plusieurs permettant principalement de rendre l'accessibilité d'une configuration ou d'un état de contrôle décidable. On peut diviser en deux catégories les stratégies que nous allons voir : les sous-approximations, où l'ensemble des configurations observé est contenu dans l'ensemble de configurations accessibles, et les sur-approximations, où l'ensemble des configurations observé contient l'ensemble des configurations du système.

Dans un premier temps, nous verrons des solutions consistant à n'autoriser que des exécutions pouvant être divisées en phases, elles-mêmes définies différemment selon les cas. Nous y verrons en particulier les systèmes k-synchronisables. Par la suite, nous verrons les systèmes avec des canaux bornés. Nous terminerons par un tour d'horizon des autres approches de la littérature.

3.2 Diviser pour mieux régner

Nous avons différentes approches par sous-approximation permettant la décidabilité de l'accessibilité dans un système communiquant en FIFO. L'une d'entre elles consiste à autoriser seulement les exécutions qui peuvent être divisées en phases, ce qui réduit donc les comportements possibles. On peut trouver dans la littérature différentes notions de phases, nous nous concentrerons tout particulièrement dans un premier temps sur les k-échanges qui permettent de définir les systèmes k-synchronisables, et dans un second temps nous verrons les autres divisions en phases présentées dans la littérature.

3.2.1 La k-synchronisabilité

Nous définissons les systèmes k-synchronisables, que nous étudierons donc principalement dans cette thèse. La définition est celle utilisée dans les travaux de Bouajjani *et al.* dans [Bouajjani et al., 2018a] auxquels nous nous comparerons régulièrement. Cette définition peut s'appliquer aussi bien aux systèmes en boîte aux lettres qu'aux systèmes en pair à pair.

Un k-échange désigne une séquence d'actions, contenant au plus k envois suivis d'au plus k réceptions, couplées aux envois précédemment faits. Dans ce chapitre, et tout le long de cette thèse, k désignera un entier tel que $k \geq 1$.

Définition 3.2.1 (k-échange)**.** Une séquence d'actions $e = a_1 \cdots a_n$ est un k-échange si $e \in \mathtt{S}^{\leq k} \cdot \mathtt{R}^{\leq k}$ telle que pour tout $j \in [1..n]$ tel que $a_j \in \mathtt{R}$, il existe $i \in [1..j[$ tel que $a_i \in \mathtt{S}$ et $a_i \vdash\!\dashv a_j$.

Un MSC est dit *k-synchrone* s'il existe une linéarisation qui est divisible en k-échanges, telle qu'un message envoyé pendant un k-échange ne peut être reçu dans un k-échange subséquent : soit la réception intervient dans le même k-échange que l'envoi, soit le message ne sera jamais reçu et est donc non couplé.

Définition 3.2.2 (k-synchrone)**.** Un MSC μ est k-synchrone si :

- il existe une linéarisation $e = e_1 \cdots e_n$ de μ où, pour tout $i \in [1..n]$, e_i est un k-échange,
- μ est réalisable.

Une exécution e est *k-synchronisable* si $msc(e)$ est k-synchrone.

On écrit $\textsc{Msc}_k(\mathcal{S})$ pour désigner l'ensemble des MSC k-synchrones :

$$\textsc{Msc}_k(\mathcal{S}) = \{msc(e) \mid e \in \textsc{Ex}(\mathcal{S}) \text{ et } msc(e) \text{ est } k\text{-synchrone}\}$$

On définit alors un système k-synchronisable comme suit :

Définition 3.2.3 (Système k-synchronisable)**.** Un système est k-synchronisable si et seulement si $\textsc{Msc}(\mathcal{S}) = \textsc{Msc}_k(\mathcal{S})$.

Exemple 3.2.1 – Le MSC de la Figure 3.1.a est 2-synchrone : le message m_1 est indépendant et constitue un k-échange à lui tout seul, tandis que les messages m_2 et m_3 ne peuvent être séparés et forment un 2-échange.

Il n'existe pas de k tel que le MSC de la Figure 3.1.b est k-synchrone. En effet, tous les messages doivent appartenir au même k-échange, mais il est impossible d'organiser les actions de

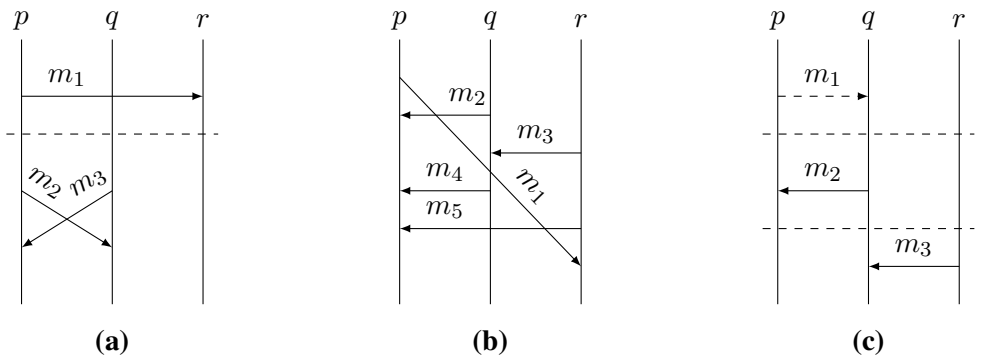


FIGURE 3.1 – Un MSC 2-synchrone (a), un MSC non k-synchrone (b) et un MSC 1-synchrone (c)

façon à avoir tous les envois suivis de toutes les réceptions, et ce notamment car la réception du message m_3 doit avoir lieu avant l'envoi du message m_4.

Soit $e_1 = \mathtt{s}(r, q, m_3) \cdot \mathtt{s}(q, p, m_2) \cdot \mathtt{s}(p, q, m_1) \cdot \mathtt{r}(q, p, m_2) \cdot \mathtt{r}(r, q, m_3)$ une exécution mb-réalisable. Son MSC est représenté en Figure 3.1.c. On peut constater que e_1 ne peut être divisée en 1-échanges. Cependant, considérons une linéarisation alternative de $msc(e_1)$:

$$e_2 = \mathtt{s}(p, q, m_1) \cdot \mathtt{s}(q, p, m_2) \cdot \mathtt{r}(q, p, m_2) \cdot \mathtt{s}(r, q, m_3) \cdot \mathtt{r}(r, q, m_3)$$

e_2 est divisible en 1-échanges dans lesquels chaque envoi appartient à un k-échange, avec la réception correspondante si elle existe. Alors, $msc(e_1)$ est 1-synchrone et e_1 est 1-synchronisable. Notez que e_2 n'est pas une exécution mb-réalisable et qu'il n'existe aucune exécution mb-réalisable correspondant à cet MSC qui serait divisible en 1-échanges. Le découpage en k-échanges souligne les dépendances entre les messages, mais n'est pas toujours corrélé aux exécutions représentées par ce MSC.

On verra, en particulier dans le Chapitre 6, qu'être k-synchronisable n'implique pas de borne sur les canaux. On peut l'apercevoir ici avec le dernier exemple où l'exécution e_1 nécessite des canaux de taille 2, bien que ce soit une exécution 1-synchronisable.

Grâce à la division en k-échanges des exécutions, l'ensemble des exécutions d'un système k-synchronisable est représentable par un automate fini où l'alphabet est l'ensemble des k-échanges. Cela permet à l'accessibilité d'un état de contrôle d'être décidable dans un système k-synchronisable quel que soit le type de communication. Nous verrons le détail de cette construction dans le Chapitre 4. Nous y verrons également qu'il est décidable de savoir si un système quelconque est k-synchronisable pour un k donné. Enfin, au Chapitre 5 nous démontrerons qu'il est décidable de savoir s'il existe un k tel que le système observé est k-synchronisable.

3.2.2 D'autres divisions en phases

Des phases, semblables aux k-échanges, formées d'une séquence d'envoi suivi d'une séquence de réceptions, sont également utilisées dans [Drăgoi et al., 2016]. Dans ces travaux, le langage PSYNC est défini et permet de construire un programme comme une suite de phases, offrant l'illusion d'une communication synchrone, autorisant la présence d'erreurs telle que la perte de message, tout en permettant la vérification des programmes ainsi décrits. On retrouve de telles phases

également dans [Chaouch-Saad et al., 2009] et [Charron-Bost and Schiper, 2009]. Dans ces cas, à chaque fin de phase, on peut considérer les canaux comme étant vides, ce qui nous permet alors de représenter les configurations comme de simples états de contrôle, et nous mène alors à un modèle fini pour représenter l'ensemble des comportements.

Parfois même, le nombre de phases dans une exécution peut être borné. L'idée est que, lorsqu'un problème intervient, c'est souvent dans les premières phases de l'exécution et il n'est donc pas nécessaire de chercher très loin dans l'évolution du système pour découvrir les erreurs. Cette technique ne permet évidemment pas de certifier l'absence d'erreur, mais permet tout de même d'augmenter la confiance que l'on peut accorder au programme. C'est le point de vue adopté dans [La Torre et al., 2007, La Torre et al., 2008, Heussner et al., 2012, Bollig et al., 2014, Bouajjani and Emmi, 2014].

Dans [La Torre et al., 2007], [La Torre et al., 2008] et [Heussner et al., 2012], une phase consiste en un ensemble d'actions d'un seul processus, il est capable alors de lire dans sa file et d'envoyer dans les autres files.

Dans [Bouajjani and Emmi, 2014], pour une phase, un nombre non borné de processus peut intervenir, pour faire un nombre non borné d'actions, cependant, chacun ne peut intervenir qu'une seule fois. Une séquence d'actions effectuée par un premier processus doit s'arrêter si elle nécessite l'intervention d'un second processus. La phase est alors terminée pour le premier processus et ses prochaines actions appartiendront à la prochaine phase.

3.3 Restriction sur l'usage des canaux

Nous nous intéresserons ici aux sous-approximations liées à la borne des canaux. Nous verrons donc en détail les systèmes bornés où les canaux ont une limite sur le nombre de messages qu'ils peuvent contenir. Une variante consiste à se concentrer sur les systèmes où toute exécution peut être simulée par une exécution k-bornée. Nous verrons également aussi une sous-classe de ceux-ci, où toute exécution est équivalente à une exécution synchrone.

3.3.1 Les systèmes bornés

Nous pouvons alors établir une borne sur les canaux sauf que celle-ci ne tient compte que des messages couplés, les messages non couplés sont ignorés et ne sont pas comptés. Si une exécution est k-bornée et ne contient pas de messages non couplés, elle est alors réalisable avec des canaux bornés de taille k. On définit alors une exécution *k-bornée* comme une exécution où le nombre de messages couplés est à tout moment inférieur ou égale à k dans chaque canal. Ainsi, formellement :

Définition 3.3.1 (Séquence d'actions k-pp-bornée, [Kuske and Muscholl, 2010])**.** Soit e une séquence d'actions. e est dite k-pp-bornée, si et seulement si $\forall p, q \in \mathbb{P}$ et $\forall v$ préfixe de e,

$$min\{\sharp(v, \mathtt{s}(p, q, \_)), \sharp(e, \mathtt{r}(p, q, \_))\} - \sharp(v, \mathtt{r}(p, q, \_)) \leq k$$

On soustrait le nombre de réceptions dans le préfixe au nombre d'envois s'il est plus petit que le nombre de réceptions totales de la séquence d'actions. Ainsi, les messages non couplés et stockés dans les canaux seront ignorés. Nous pouvons adapter cette définition aux systèmes communiquant en boîte aux lettres.

Définition 3.3.2 (Séquence d'actions k-mb-bornée)**.** Soit e une séquence d'actions. e est dite k-mb-bornée, si et seulement si $\forall q \in \mathbb{P}$ et $\forall v$ préfixe de e,

$$min\{\sharp(v, \mathtt{s}(\_, q, \_)), \sharp(e, \mathtt{r}(\_, q, \_))\} - \sharp(v, \mathtt{r}(\_, q, \_)) \leq k$$

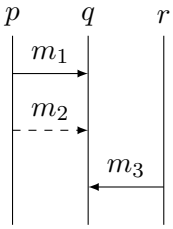
On définit alors $\text{Ex}_k(\mathcal{S}) = \{e \mid e \in \text{Ex}(\mathcal{S}), e \text{ est } k\text{-bornée}\}$ comme l'ensemble des exécutions k-bornées du système $\mathcal{S}$. Dans la suite de cette thèse, lorsque nous parlerons de canaux bornés à k, il faudra comprendre "contenant au plus k messages couplés à tout moment".

Une variante des exécutions k-pp-bornées peut être trouvée dans [Genest et al., 2007], toujours pour les systèmes en pair à pair. Dans celle-ci, seules les configurations stables et donc, les exécutions ne contenant pas de messages non couplés sont considérées dans le système. Cette définition est étendue aux systèmes en boîte aux lettres dans [Bollig et al., 2021].

Exemple 3.3.1 – La Figure 3.2 représente un MSC et certaines linéarisations possibles. Chacune de ces linéarisations a ses spécificités. La linéarisation e_1 est 1-pp-bornée : à chaque instant, chaque canal ne contient au plus qu'un message couplé. Enfin, cette linéarisation est mb-réalisable, et on peut également constater qu'elle est 1-mb-bornée.

Pour la linéarisation e_2, dans une communication en pair à pair, il n'y a également qu'un message dans chaque canal au maximum à chaque moment, elle est donc 1-pp-bornée. Cependant, si l'on considère une communication en boîte aux lettres, alors les messages m_2 et m_3 sont, à un moment donné, tous les deux dans le canal c_q, e_2 est donc 2-mb-bornée.

Enfin, la linéarisation e_3, dans un contexte en pair à pair, les canaux doivent être de taille 2 pour les messages m_1 et m_2 qui sont stockés au même moment dans le canal c_q, e_3 est donc 2-pp-bornée. Enfin, cette linéarisation n'est pas mb-réalisable, le message m_3 étant envoyé au processus q avant le message m_1 mais étant lu après : ce comportement ne respecterait pas les canaux FIFO en boîte aux lettres.



$$e_1 = \mathtt{s}(p, q, m_1) \cdot \mathtt{r}(p, q, m_1) \cdot \mathtt{s}(r, q, m_3) \cdot \mathtt{r}(r, q, m_3) \cdot \mathtt{s}(p, q, m_2)$$

$$e_2 = \mathtt{s}(p, q, m_1) \cdot \mathtt{r}(p, q, m_1) \cdot \mathtt{s}(r, q, m_3) \cdot \mathtt{s}(p, q, m_2) \cdot \mathtt{r}(r, q, m_3)$$

$$e_3 = \mathtt{s}(r, q, m_3) \cdot \mathtt{s}(p, q, m_1) \cdot \mathtt{s}(p, q, m_2) \cdot \mathtt{r}(p, q, m_1) \cdot \mathtt{r}(r, q, m_3)$$

FIGURE 3.2 – Un MSC et certaines de ses linéarisations

À partir de cette définition, on peut alors caractériser les MSC bornés. L'idée est que toutes les exécutions correspondantes et com-réalisables y sont k-com-borné avec com étant la communication du système. Cependant, cette contrainte est très restrictive et n'est pas très vraisemblable. Pour être plus réaliste, on peut simplement demander que le MSC admette au moins une linéarisation qui soit com-réalisable et k-com-borné. Ainsi, dans un tel système, chaque exécution a une exécution k-com-borné équivalente, qui a donc le même MSC. On pourra alors distinguer les MSC dont toutes les linéarisations sont bornées, et les MSC dont au moins une linéarisation est bornée. On les appellera respectivement *universellement* et *existentiellement* bornés. Plus formellement, si toutes les linéarisations d'un MSC sont bornées par le même k, on parle d'un MSC universellement borné. Si au moins une linéarisation du MSC est k-bornée, alors, on parle d'un MSC existentiellement k-borné.

Définition 3.3.3 (MSC universellement borné, MSC existentiellement borné)**.** Un MSC μ est dit universellement k-com-borné, noté $\forall$-k-com-borné, pour com $\in \{\mathsf{mb}, \mathsf{pp}\}$, si et seulement si toutes les linéarisations sont k-com-bornées.

Un MSC μ est dit existentiellement k-com-borné, noté $\exists$-k-com-borné, pour com $\in \{\mathsf{mb}, \mathsf{pp}\}$, si et seulement s'il existe une linéarisation k-com-bornée.

Exemple 3.3.2 – Nous pouvons à présent caractériser le MSC de la Figure 3.2. Compte tenu de la caractérisation de ses linéarisations, on peut affirmer qu'il est $\exists$-1-pp-borné, mais seulement $\forall$-2-pp-borné. De la même façon, ce MSC est $\exists$-1-mb-borné et $\forall$-2-mb-borné.

Enfin, ces définitions sont également étendues aux systèmes.

Définition 3.3.4 (Système universellement borné, système existentiellement borné)**.** Soit un système $\mathcal{S} = ((L_p, \delta_p, \ell_p^0)_{p\in\mathbb{P}}, \mathsf{com})$.

$\mathcal{S}$ est $\forall$-k-com-borné si, pour tout MSC $\mu \in \textsc{Msc}(\mathcal{S})$, μ est $\forall$-k-com-borné.

$\mathcal{S}$ est $\exists$-k-com-borné si, pour tout MSC $\mu \in \textsc{Msc}(\mathcal{S})$, μ est $\exists$-k-com-borné.

Exemple 3.3.3 – Supposons un système $\mathcal{S}$ composé des processus p et q décrits dans la Figure 3.3.a. Ainsi, les MSC μ_1 et μ_2 des Figures 3.3.b et 3.3.c appartiennent à l'ensemble $\textsc{Msc}(\mathcal{S})$. On constate que μ_1 est $\forall$-1-pp-borné et que μ_2 est $\forall$-2-pp-borné. Cependant, chaque MSC du système a une borne différente, et, pour tout k, on peut trouver un MSC qui ne soit pas $\exists$-k-pp-borné. Alors $\mathcal{S}$ n'est ni existentiellement ni universellement borné, pour aucun k.

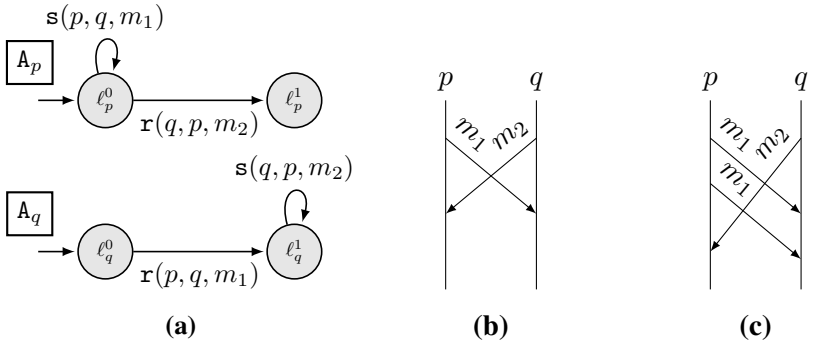


FIGURE 3.3 – Système $\mathcal{S}$ (a) et les MSC μ_1 (b) et μ_2 (c)

Le problème de l'accessibilité dans un système k-borné est décidable et constitue une sous-approximation du problème de l'accessibilité dans un système d'automates communicants. Notamment, l'accessibilité d'un état de contrôle est décidable lorsque l'on travaille sur un système $\forall$-k-pp-borné ou $\exists$-k-pp-borné. En effet, comme démontré dans [Kuske and Muscholl, 2010, Proposition 7.1], le problème revient à l'accessibilité d'un état de contrôle dans un système où le nombre de configurations est fini, ce qui est donc décidable.

Cependant, il n'est pas toujours possible de savoir si un système est borné. Regardons les résultats pour les systèmes en pair à pair. Si aucun k n'est donné en paramètre, le problème de savoir si un système est universellement ou existentiellement borné est en général indécidable ([Kuske and Muscholl, 2010, Propositions 7.3.(2)]). Savoir, pour un k donné, si un système est universellement k-borné (respectivement existentiellement k-borné) est décidable, que ce soit pour

la sémantique pair à pair [Kuske and Muscholl, 2010, Propositions 7.3.(1)] ou pour la sémantique boîte aux lettres [Bollig et al., 2021, Proposition 38] *.

3.3.2 Les systèmes synchronisables

Une classe proche des systèmes $\exists$-k-pp-borné est définie dans [Basu and Bultan, 2016] sous le nom de systèmes *k-stables*, et dans des cas particuliers, les systèmes *synchronisables*. L'objectif est encore une fois de borner les canaux afin de réduire l'ensemble de recherche, et permettant la décidabilité de certains problèmes. Ici, ce sont les traces d'envois des exécutions qui sont étudiées, c'est-à-dire les projections des exécutions, restreintes aux envois. Si l'ensemble des traces d'envois des exécutions synchrones est identique à l'ensemble des traces d'envois des exécutions k-bornées, on dit alors que le système est k-stable. S'il est identique à l'ensemble des traces d'envois des exécution asynchrones, on dit que que le système est synchronisable.

Le problème d'être synchronisable pour un système donné, communiquant en boîte aux lettres ou en pair à pair, était montré comme décidable dans [Basu and Bultan, 2016] avant que Finkel et Lozes apportent deux contre-exemples dans [Finkel and Lozes, 2017] montrant que la preuve établie était fausse pour ces deux types de communication. En effet, la preuve s'appuyait sur un lemme affirmant que si l'ensemble des traces d'envois synchrones était égale à l'ensemble des traces d'envois 1-bornées, alros le système était synchronisable. Les contre-exemples de [Finkel and Lozes, 2017] exposent des cas où l'ensemble des traces d'envois 2-bornées contient des traces n'existant pas dans l'ensemble des traces d'envois 1-bornées pourtant identique à l'ensemble des traces d'envois synchrones. De plus, l'indécidabilité de l'appartenance pour les systèmes communiquant en pair à pair est apportée dans [Finkel and Lozes, 2017, Théorème 3] tandis que le problème reste ouvert pour les systèmes communiquant en boîte aux lettres.

3.4 D'autres approches

Dans la littérature, d'autres approches ont été étudiées. Bien que nous n'y reviendrons pas dans les travaux effectués dans cette thèse, en voici certaines.

3.4.1 Des approches par sous-approximation

Restriction sur l'usage des canaux

Une stratégie consiste à imposer une contrainte sur le fonctionnement des canaux. C'est le cas avec les systèmes avec canaux à l'alternat (*half duplex*) où les deux canaux du système ne sont pas bornés, mais seul un canal peut être utilisé à la fois. Plus exactement, toute configuration accessible du système a au plus un canal non vide. Les système à alternats ont été définis dans [Cécé and Finkel, 2005]. Dans ce papier il est établi que l'accessibilité d'un état de contrôle pour un système à alternat, ainsi que le fait d'être un système à alternats sont deux propriétés décidables. L'appartenance d'un système à la classe des systèmes à l'alternat y est elle aussi prouvée décidable. De plus, l'ensemble des configurations d'un tel système est régulier ce qui facilite la vérification des problèmes de sûreté, notamment l'absence d'interblocage. Cette classe a également

*. Attention, dans [Kuske and Muscholl, 2010] la notion d'automate communicant comporte des états acceptants, elle est donc différente de celle utilisée dans cette thèse. Les systèmes communicants tels que définis dans cette thèse sont des systèmes "deadlock-free" dans la terminologie de [Kuske and Muscholl, 2010].

été étendue aux systèmes composés de plus de deux machines. Dans [Di Giusto et al., 2021a], l'approche des auteurs pour étendre cette sous-classe donne lieu à des systèmes qui ne sont pas Turing-équivalent et qui permettent alors la décidabilité de nombreux problèmes de vérification, dont l'accessibilité d'une configuration et en particulier d'une configuration d'interblocage.

Restriction sur le graphe de contrôle

Une autre solution est d'attribuer des propriétés particulières aux canaux. Si un langage des canaux, c'est-à-dire l'ensemble des mots que constituent les contenus de canaux, peut être reconnu par un automate fini, autrement dit, il s'agit d'un langage régulier, alors on dit que le langage des canaux est reconnaissable. Dans ce cas, l'accessibilité d'une configuration est décidable, comme démontré dans [Pachl, 2012]. Notez que si l'on considère un système où un processus envoie les mêmes suites de messages à deux autres processus, le langage serait de la forme $a^n b^n$ et ne sera donc pas régulier.

Si le langage des canaux est reconnaissable, on peut alors représenter un système, comme dans [Boigelot et al., 1997] ou [Boigelot and Godefroid, 1999], par un diagramme de décision sur le contenu des files d'attente (QDD), un automate formé d'un ensemble de configurations en guise d'ensemble d'états et où les transitions correspondent à des contenus de canaux.

Différentes variantes de cette restriction sur les canaux ont donné naissance à différentes classes comme les machines bornées en entrée dans [Bollig et al., 2020], pour lesquelles l'accessibilité d'un état de contrôle est décidable. On peut trouver des études sur d'autres classes similaires telles que les machines bornées en entrée par les lettres dans [Gouda et al., 1987], les réseaux FIFO monogènes dans [Finkel and Schnoebelen, 2001] et [Finkel, 1990], les FIFO linéaires [Choquet and Finkel, 1987, Finkel, 1990] et [Jéron and Jard, 1993] ou encore les machines "plates" étudiées dans [Finkel and Praveen, 2019] et [Esparza et al., 2012].

3.4.2 Des approches par sur-approximation

Sémantique avec insertion de messages

Une première sur-approximation que l'on peut faire est de supposer que les canaux peuvent faire des erreurs. Ces erreurs peuvent être le gain de message avec des machines à canaux avec insertion (*insertion channels machines*), où les canaux peuvent acquérir du contenu additionnel spontanément, comme étudiés dans [Bouyer et al., 2012, Cécé et al., 1996]. Pour ces systèmes, l'accessibilité d'un état de contrôle est prouvée décidable [Ouaknine and Worrell, 2005].

Sémantique avec perte de messages

Ces erreurs peuvent aussi être la perte de messages avec des machines à canaux avec pertes (*lossy channel machines*), comme étudiés dans [Abdulla et al., 2012], où les messages peuvent disparaître de façon non deterministe des canaux sans avoir été lus.

Les machines à canaux avec pertes sont des systèmes de transition bien structurés (*Well Structure Transition Systems*), étudiés notamment par Finkel *et al.* et Abdulla *et al.* dans [Finkel and Schnoebelen, 2001, Choquet and Finkel, 1987, Abdulla and Jonsson, 1996, Abdulla et al., 1996], où l'ensemble des configurations est doté d'un bel ordre, ce qui permet la décidabilité de propriétés intéressantes telles que l'accessibilité d'un état de contrôle, prouvée dans [Abdulla and Jonsson, 1996], ou la terminaison, prouvée dans

[Finkel and Schnoebelen, 2001]. Ces résultats sont également valables pour les systèmes à canaux avec pertes [Abdulla et al., 1996], constituant une sous-classe des WSTS.

Sémantique non ordonnée

Concernant les messages dans les canaux, l'on peut aussi ignorer l'ordre des messages dans les canaux, comme developpé dans le survey [Kuske and Muscholl, 2010], ce qui rend notre système équivalent à un réseau de Pétri, auquel cas l'accessibilité devient décidable, mais la notion de FIFO est alors perdue.

3.5 Conclusion

Ce chapitre nous a donc permis de définir les problèmes d'accessibilité que nous étudierons dans cette thèse. Nous avons également vu une partie des approximations possibles afin de rendre ces problèmes décidables. Nous avons alors défini les systèmes k-synchronisables qui seront au cœur de nos travaux dans les Chapitres 4 et 5. Plus précisémment, le Chapitre 4 apportera une preuve de décidabilité de l'accessibilité d'un état de contrôle et de la décidabilité de l'appartenance à la classe des systèmes k-synchronisables pour un k donné. Le Chapitre 5 apportera la preuve de l'on peut trouver le k tel qu'un système donné est k-synchronisable, s'il existe. Nous avons également défini les systèmes k-bornés qui nous seront utiles à des fins de comparaisons dans le Chapitre 6.

Contributions

Chapitre 4

La décidabilité de la k-synchronisabilité pour un k donné

La k-synchronisabilité, définie dans la Section 3.2.1 du Chapitre 3 fut introduite et étudiée par Bouajjani *et al.* dans [Bouajjani et al., 2018a], dont une version longue est disponible [Bouajjani et al., 2018b].

Ces travaux établissent la décidabilité de la k-synchronisabilité pour un k donné et un système donné, ainsi que la décidabilité de l'accessibilité d'un état de contrôle dans un système k-synchronisable. Cependant, nous y avons relevé quelques anomalies qui nous ont poussés à aller plus loin et à apporter de nouvelles preuves, s'appuyant sur les précédentes tout en les rectifiant. C'est ce que nous présenterons dans ce chapitre, nous comparerons tout du long nos preuves à celles présentées dans [Bouajjani et al., 2018a].

Ainsi, savoir si un système est k-synchronisable pour un k donné est un problème décidable. Toutefois, la preuve apportée dans [Bouajjani et al., 2018a] n'est pas suffisante. Des ambiguïtés résident dans le lien existant entre la livraison causale et les exécutions du systèmes. Elles sont dues à la communication en boîte aux lettres. Une erreur est également présente dans la caracterisation graphique des exécutions k-synchronisables, que nous détaillerons dans la Comparaison 1. Ambiguïtés et erreur associées, la preuve de décidabilité d'accessibilité d'un état de contrôle omet certains cas.

La décidabilité de l'accessibilité étant utilisée pour prouver la décidabilité de l'appartenance à la classe des systèmes k-synchronisables, ces erreurs se répercutent. D'autres inexactitudes, liées encore une fois à la communication en boîte aux lettres et à la livraison causale, détaillées cette fois-ci dans les Comparaisons 3 et 4, apparaissent cette fois-ci dans la preuve de décidabilité de la k-synchronisabilité. Ajoutées aux précédentes, il en découle que les résultats donnés par l'algorithme peuvent contenir des faux négatifs et des faux positifs.

Pour les corriger, nous utiliserons la notion de réalisabilité * en lieu et place de la livraison causale. Nous étaierons ce choix dans la Comparaison 2. Nous présenterons alors une nouvelle preuve pour la décidabilité de l'accessibilité, mais également pour la décidébilité de la k-synchronisabilité pour un k donné, en comblant les lacunes présentes dans [Bouajjani et al., 2018a] concernant la communication en boîte aux lettres.

Ce chapitre s'articule comme suit. Dans un premier temps, nous nous intéresserons aux systèmes communiquant en boîte aux lettres, la Section 4.1 décrira des caractérisations graphiques

$*$. Dans [Di Giusto et al., 2020] et [Di Giusto et al., 2021b], nous conservons le nom de *causal delivery* tout en modifiant la définition pour celle de la réalisabilité.

pour les MSC réalisables ainsi que pour les MSC k-synchrones. La preuve de la décidabilité de l'accessibilité suivra dans la Section 4.2, utilisant cette première caractérisation. Enfin, la preuve de décidabilité de la k-synchronisabilité utilisera la seconde caractérisation dans la Section 4.3. La Section 4.4 adaptera les sections précédentes aux systèmes communiquant en pair à pair, prouvant ainsi la décidabilité de la k-synchronisabilité pour les systèmes avec ce type de communication également.

4.1 Caractérisations graphiques

Le graphe de dépendances d'un MSC, défini par la Définition 2.2.2 dans le Chapitre 2 permet non seulement de mettre en évidence les dépendances qu'il représente, mais il peut aussi nous permettre de déterminer si le MSC auquel il est associé est réalisable ou encore s'il est k-synchrone.

4.1.1 Un MSC réalisable

La réalisabilité en boîte aux lettres d'un MSC peut être caractérisée graphiquement. En effet, les arcs du graphe de dépendances donnent un ordre nécessaire à respecter entre les actions pour obtenir une linéarisation mb-réalisable. Cependant, cet ordre partiel est nécessaire, mais pas suffisant.

Si deux messages, l'un couplé et l'autre non, sont envoyés au même destinataire, le message couplé doit être envoyé avant le message non couplé. Si ce n'est pas le cas, le message non couplé sera en tête de file, mais, sa lecture ne faisant pas partie de la linéarisation, il y restera et empêchera le message couplé d'être lu. Donc on peut déduire un ordre sur les envois de ces messages. D'autre part, si deux messages sont reçus dans un certain ordre, par un même destinataire, alors on sait que ces messages sont entrés dans la file dans ce même ordre et l'on peut encore une fois déduire un ordre sur les envois. Enfin, si l'on sait qu'une action doit avoir lieu avant une autre qui elle-même doit avoir lieu avec une troisième action, alors, on peut déduire que la première doit avoir lieu avant la troisième et un arc pourrait représenter cet ordre.

Avec ces idées, l'on peut construire de nouveaux arcs qui constitueront le *graphe de dépendances étendu*. Celui-ci est composé des sommets du graphe de dépendances liés par des *arcs étendus*. Ceux-ci sont déduits par des raisonnements dont ceux cités ci-dessus. On ajoute à ca que, pour un message couplé, on sait que son envoi a lieu avant sa réception, et que d'un arc du graphe de dépendances l'on peut déduire un arc étendu.

Alors une linéarisation doit respecter non seulement l'ordre induit par les arcs du graphe de dépendances, équivalent à l'ordre partiel du MSC, mais également l'ordre induit par les arcs étendus du graphe de dépendances étendu. Néanmoins, si l'on sait que l'ordre induit par le MSC est irréflexif, ce n'est pas forcément le cas de l'ordre que nous donne ces arcs étendus. Cela se traduirait par un cycle étiqueté SS, ce qui implique que l'envoi d'un message doit avoir lieu avant son envoi. Cette incohérence indique alors que le MSC n'est pas mb-réalisable puisqu'on ne peut construire aucune linéarisation mb-réalisable.

Ainsi, formellement, nous introduisons les arcs étendus notés $\mathbf{m} \overset{\mathrm{XY}}{\dashrightarrow} \mathbf{m}'$. La relation $\mathbf{m} \overset{\mathrm{XY}}{\dashrightarrow} \mathbf{m}'$ est définie dans la Figure 4.1 avec X,Y $\in \{$S,R$\}$. $\mathbf{m} \overset{\mathrm{XY}}{\dashrightarrow} \mathbf{m}'$ désigne le fait que l'événement X du message $\mathbf{m}$ doit se produire avant l'événement Y du message $\mathbf{m}'$, ceci dû soit à l'ordre de leurs actions sur la même machine (RÈGLE 1), soit au fait qu'un envoi couplé doit avoir lieu avant sa réception (RÈGLE 2), soit à des contraintes de la sémantique boîte aux lettres (RÈGLES 3

$$\text{(RÈGLE 1)}\ \frac{\mathbf{m}_1 \xrightarrow{\text{XY}} \mathbf{m}_2}{\mathbf{m}_1 \overset{\text{XY}}{\dashrightarrow} \mathbf{m}_2} \qquad \text{(RÈGLE 2)}\ \frac{\mathbf{m} \cap \mathtt{R} \neq \emptyset}{\mathbf{m} \overset{\text{SR}}{\dashrightarrow} \mathbf{m}} \qquad \text{(RÈGLE 3)}\ \frac{\mathbf{m}_1 \xrightarrow{\text{RR}} \mathbf{m}_2}{\mathbf{m}_1 \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_2}$$

$$\text{(RÈGLE 4)}\ \frac{\mathbf{m}_1 \cap \mathtt{R} \neq \emptyset \qquad \mathbf{m}_2 \cap \mathtt{R} = \emptyset \qquad \mathsf{proc}_{\mathtt{R}}(\mathbf{m}_1) = \mathsf{proc}_{\mathtt{R}}(\mathbf{m}_2)}{\mathbf{m}_1 \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_2} \qquad \text{(RÈGLE 5)}\ \frac{\mathbf{m}_1 \overset{\text{XY}}{\dashrightarrow} \overset{\text{YZ}}{\dashrightarrow} \mathbf{m}_2}{\mathbf{m}_1 \overset{\text{XZ}}{\dashrightarrow} \mathbf{m}_2}$$

FIGURE 4.1 – Règles de déduction pour les arcs de dépendances étendus

et 4), ou encore à une succession de telles dépendances (RÈGLE 5). Un cycle $\mathbf{m} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}$ apparaît si et seulement si le MSC n'est pas mb-réalisable.

Exemple 4.1.1 – La Figure 4.2.a représente un MSC et son graphe de dépendances étendu est représenté en Figure 4.2.b. Cet MSC n'est pas mb-réalisable : en effet, les messages m_1 et m_3 sont envoyés au même processus q, m_3 est couplé, mais m_1 ne sera jamais reçu. Ainsi, on sait que m_3 doit être envoyé avant m_1. Cependant, pour envoyer m_3, on a besoin de recevoir m_2 et donc d'envoyer m_1, on ne sera donc pas capable de trouver une linéarisation de cet MSC mb-réalisable.

On peut faire le même raisonnement en regardant le graphe de dépendances et en appliquant les règles de la Figure 4.1. On peut commencer par appliquer la RÈGLE 2 au sommet $\mathbf{m}_2$ et donc ajouter l'arc étendu SR. Grâce à cet arc et la RÈGLE 5 (et la RÈGLE 1, de façon implicite), on peut ajouter un arc SS entre $\mathbf{m}_2$ et $\mathbf{m}_3$. Ce dernier nous permet de déduire, avec la RÈGLE 5, l'arc SS de $\mathbf{m}_1$ à $\mathbf{m}_3$. Finalement, la RÈGLE 4 nous permet d'ajouter un arc SS depuis $\mathbf{m}_3$ à $\mathbf{m}_1$. Les deux derniers arcs étendus associés à la RÈGLE 5 nous amènent à créer une boucle sur $\mathbf{m}_1$ étiquetée SS, indiquant que $\mathbf{m}_1$ doit être envoyé avant $\mathbf{m}_1$, ce qui signifie que ce MSC n'est pas mb-réalisable.

On constate que les arcs présents entre $\mathbf{m}_1$ et $\mathbf{m}_3$, dans un sens comme dans l'autre, n'existaient pas dans le graphe de dépendances du MSC, mais ces dépendances apparaissent grâce au graphe de dépendances étendu.

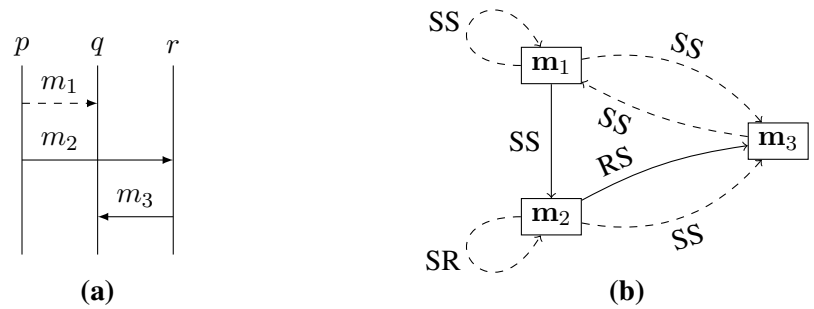


FIGURE 4.2 – Un MSC non mb-réalisable et son graphe de dépendances étendu

Nous pouvons donc établir le théorème suivant pour une caractérisation graphique d'un MSC mb-réalisable.

Théorème 4.1.1. *Un MSC est* mb*-réalisable si et seulement s'il n'existe pas de cycle de la forme* $\mathbf{m} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}$ *dans son graphe de dépendances étendu pour un certain sommet* $\mathbf{m}$.

Démonstration.

$\Rightarrow$ Soit μ un MSC mb-réalisable. Alors, il existe un ordre total $<$ sur les événements qui est une linéarisation de $\prec = (\prec_{po} \cup \prec_{src})^+$ (cf. Définition 2.2.3) avec la propriété établie dans la Définition 2.2.5. On affirme que si $\mathbf{m} \overset{\text{XY}}{\dashrightarrow} \mathbf{m}'$, $\{a_i\} = \mathbf{m} \cap \mathtt{X}$ et $\{a_j\} = \mathbf{m}' \cap \mathtt{Y}$, alors $i < j$. La preuve de cette affirmation se fait par récurrence sur l'arbre de dérivation de $\mathbf{m} \overset{\text{XY}}{\dashrightarrow} \mathbf{m}'$:

- cas de la RÈGLE 1 : $(i,j) \in \prec_{po}$ donc $i < j$;
- cas de la RÈGLE 2 : $(i,j) \in \prec_{src}$ donc $i < j$;
- cas des RÈGLES 3 et 4 : par définition de la réalisabilité en boîte aux lettres ;
- cas de la RÈGLE 5 : il existe $\mathbf{m}_3$ tel que $\mathbf{m}_1 \overset{\text{XZ}}{\dashrightarrow} \mathbf{m}_3 \xrightarrow{\text{ZY}} \mathbf{m}_2$. On pose a_l l'action qui appartient à $\mathbf{m}_3 \cap \mathtt{Z}$. Par hypothèse d'induction, $i < l < j$, et par transitivité de $<$, $i < j$.

Ainsi, nous prouvons notre affirmation et $<$ étend $\overset{\text{XY}}{\dashrightarrow}$. Par conséquent, il n'existe pas de cycle $\overset{\text{SS}}{\dashrightarrow}$.

$\Leftarrow$ Supposons que le graphe de dépendances étendu ne contient pas de cycle $\overset{\text{SS}}{\dashrightarrow}$.

Montrons tout d'abord qu'il ne contient pas de cycle $\overset{\text{RR}}{\dashrightarrow}$ non plus. Par contradiction, supposons qu'il existe un sommet $\mathbf{m}$ tel quel $\mathbf{m} \overset{\text{RR}}{\dashrightarrow} \mathbf{m}$. Puisque qu'il n'existe pas de cycle $\overset{\text{SS}}{\dashrightarrow}$, il n'y a pas de message $\mathbf{m}'$ dans ce chemin cyclique tel que $\mathbf{m} \overset{\text{RS}}{\dashrightarrow} \mathbf{m}' \overset{\text{SR}}{\dashrightarrow} \mathbf{m}$. Donc, $\mathbf{m}(\xrightarrow{\text{RR}})^*\mathbf{m}$, et on atteint une contradiction car, $\xrightarrow{\text{RR}}$ est inclue dans $\prec_{po}$ qui est acyclique. Ainsi, $\overset{\text{RR}}{\dashrightarrow}$ est acyclique et $\overset{\text{XY}}{\dashrightarrow}$ définit un ordre partiel sur les actions.

Prenons une linéarisation de cet ordre, et nommons $<$ l'ordre associé sur les indices, c'est-à-dire, $<$ est un ordre total tel que pour toute $\mathtt{X}$ type d'action de $a_i \in \mathbf{m}_i$ et $\mathtt{Y}$ type d'action de $a_j \in \mathbf{m}_j$, $\mathbf{m}_i \overset{\text{XY}}{\dashrightarrow} \mathbf{m}_j$ implique $i < j$. On veut montrer que $<$ satisfait la propriété de la Définition 2.2.5. Soit $i < j$ avec $a_i, a_j \in \mathtt{S}$ et $\mathsf{proc}_{\mathtt{R}}(a_i) = \mathsf{proc}_{\mathtt{R}}(a_j)$ et soient $\mathbf{m}_i, \mathbf{m}_j$ deux sommets tels que $a_i \in \mathbf{m}_i$ et $a_j \in \mathbf{m}_j$. Puisque $<$ étend $\overset{\text{XY}}{\dashrightarrow}$, on a soit $\mathbf{m}_i \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_j$, soit $\neg(\mathbf{m}_j \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_i)$.

- Supposons que $\mathbf{m}_i \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_j$. Si $\mathbf{m}_i$ est non couplé, alors $\mathbf{m}_j$ doit être non couplé, autrement, par la RÈGLE 4 on aurait $\mathbf{m}_j \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_i$, ce qui contredirait l'hypothèse d'acyclicité. D'un autre côté, si $\mathbf{m}_i$ et $\mathbf{m}_j$ sont couplés, alors $\mathbf{m}_i \xrightarrow{\text{RR}} \mathbf{m}_j$, sinon on aurait $\mathbf{m}_j \xrightarrow{\text{RR}} \mathbf{m}_j$, et, par la RÈGLE 3, $\mathbf{m}_j \xrightarrow{\text{SS}} \mathbf{m}_j$, ce qui contredirait aussi l'hypothèse d'acyclicité. Ainsi, il y a i', j' tels que $\mathbf{m}_i = \{a_i, a_{i'}\}, \mathbf{m}_j = \{a_j, a_{j'}\}$ et $i' < j'$, comme requis par la Définition 2.2.5.

✢ Supposons que $\neg(\mathbf{m}_i \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_j)$ et $\neg(\mathbf{m}_j \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_i)$. Ainsi, les deux envois sont non couplés (à cause des RÈGLES 3 et 4), donc la propriété de la Définition 2.2.5 est validée, ceci concluant la preuve.

□

4.1.2 Un MSC k-synchrone

Cette seconde partie décrit alors une caractérisation graphique d'un MSC k-synchrone. On parlera d'une *composante fortement connexe* (*Strongly Connected Componant* en anglais) si pour toute paire de sommets $\mathbf{m}, \mathbf{m}' \in U$, $U \subseteq V$ un ensemble de sommets, pour un graphe donné (V, E), il existe deux chemins orientés $\mathbf{m} \rightarrow^* \mathbf{m}'$ et $\mathbf{m}' \rightarrow^* \mathbf{m}$. On peut alors caractériser un MSC k-synchrone à l'aide des composantes fortement connexes du graphe de dépendances associé.

Dans un graphe de dépendances, une composante fortement connexe indique un lien entre chaque paire de messages via leur action. Ainsi, ces connexions impliquent que ces messages doivent appartenir au même k-échange. La présence d'un arc RS dans une composante fortement connexe indique qu'une réception doit être faite pour qu'un envoi puisse avoir lieu, ce qui empêche donc d'organiser les envois suivis des réceptions. Approfondissons cette intuition avec plusieurs exemples de MSC et de leur graphe de dépendances, et faisons le lien entre leur k-synchronisabilité et ces derniers.

Exemple 4.1.2 – Dans le MSC de la Figure 4.3.a, les 4 messages présents ne peuvent être séparés et appartiendraient donc au même k-échange. Cependant, on ne peut pas organiser les actions de façon à structurer le k-échange comme il se doit : envois suivis de réception. En effet, on constate que la réception de $\mathbf{m}_2$ est nécessaire à l'envoi de $\mathbf{m}_3$. Ainsi, cet MSC n'est pas k-synchrone quelque soit le k.

Regardons à présent son graphe de dépendances en Figure 4.3.b. On constate que tous les messages appartiennent à la même composante fortement connexe, ce qui représente le fait qu'ils doivent appartenir au même k-échange, qui serait donc de taille 4. Cependant, on retrouve une arête RS entre $\mathbf{m}_2$ et $\mathbf{m}_3$ qui représente le fait que la réception de $\mathbf{m}_2$ doit précéder l'envoi de $\mathbf{m}_3$ et qui démontre alors que l'organisation en k-échange est impossible, ce qui rend ce MSC non k-synchrone quelque soit le k.

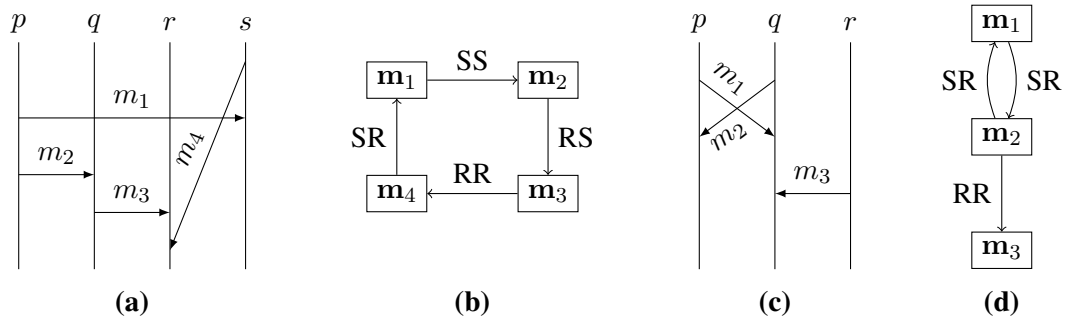


FIGURE 4.3 – Des MSC et leurs graphes de dépendances

Le MSC de la Figure 4.3.c est, quant à lui, 2-synchrone. En effet, le plus grand k-échange de cet MSC regroupe $\mathbf{m}_1$ et $\mathbf{m}_2$, qui comme on le constate sur le graphe de dépendances en

Figure 4.3.d, doivent appartenir au même k-échange, puisqu'appartenant à la même composante fortement connexe. L'absence d'arc RS assure le fait de pouvoir organiser les actions dans ce k-échange afin d'avoir tous les envois suivis par toutes les réceptions.

Ainsi, chaque composante fortement connexe regroupe les messages qui doivent appartenir au même k-échange et l'absence d'arc RS au sein d'un k-échange assure le fait de pouvoir l'organiser de sorte que tous les envois précèdent les réceptions. Nous pouvons alors établir le théorème suivant.

Théorème 4.1.2. *Soit μ un MSC* mb*-réalisable. μ est k-synchrone si et seulement si toute composante fortement connexe du graphe de dépendances est de taille inférieure ou égale à k et ne contient pas d'arc RS.*

Démonstration.

Soit μ un MSC mb-réalisable.

$\Rightarrow$ Si μ est k-synchrone, alors $\exists e = e_1 \cdots e_n$ tel que $msc(e) = \mu$ où chaque e_i est un k-échange. Pour chaque sommet $\mathbf{m}$ du graphe de dépendances $\mathsf{GD}(e)$ il y a exactement un indice $\iota(\mathbf{m}) \in [1..n]$ tel que $\mathbf{m} \subseteq e_{\iota(\mathbf{m})}$. Maintenant, observons que, s'il y a un arc de $\mathbf{m}$ à $\mathbf{m}'$ dans le graphe de dépendances, une des actions de $\mathbf{m}$ doit avoir lieu avant une action de $\mathbf{m}'$, c'est-à-dire, $\iota(\mathbf{m}) \leq \iota(\mathbf{m}')$. Donc, si $\mathbf{m}, \mathbf{m}'$ sont dans la même composante fortement connexe, alors $\iota(\mathbf{m}) = \iota(\mathbf{m}')$ et ils doivent apparaître dans le même k-échange. Puisqu'un k-échange contient au plus k messages, cela montre que toute composante fortement connexe est de taille au plus k. De plus, si $\mathbf{m} \xrightarrow{\text{RS}} \mathbf{m}'$, alors $\iota(\mathbf{m}) < \iota(\mathbf{m}')$, puisque dans un k-échange tous les envois doivent précéder les réceptions. Ainsi, une arête RS ne peut apparaître dans un cycle.

$\Leftarrow$ Soit e une linéarisation de μ. Supposons que le graphe de dépendances $\mathsf{GD}(e)$ ne contient aucune composante fortement connexe de taille supérieure à k ou contenant un arc RS. Soit $V_1, \cdots, V_n$ l'ensemble des composantes fortement connexes maximales du graphe de dépendances, listées dans un ordre topologique. Pour un indice i donné, soit $e_i = s_1 \cdots s_m r_1 \cdots r_{m'}$ l'énumération des actions des messages de V_i, définie en prenant d'abord tous les envois de V_i dans l'ordre dans lequel ils apparaissent dans e, puis les réceptions de V_i dans le même ordre que dans e aussi. Posons $e' = e_1 \cdots e_n$, alors $\mathsf{GD}(e')$ est le même que $\mathsf{GD}(e)$: en effet, la permutation d'actions que nous définissons peut seulement retarder une réception après un envoi de la même composante fortement connexe, donc il peut seulement remplacer un arc $\mathbf{m} \xrightarrow{\text{RS}} \mathbf{m}'$ par un arc $\mathbf{m}' \xrightarrow{\text{SR}} \mathbf{m}$ entre deux sommets $\mathbf{m}, \mathbf{m}'$ d'une même composante fortement connexe. Mais nous supposons qu'il n'y a pas d'arc RS dans aucune composante fortement connexe, et donc ceci ne peut arriver. Ainsi, e et e' ont le même graphe de dépendances et $msc(e') = msc(e)$. De plus, aussi par hypothèse, $\mid V_i \mid \leq k$ pour tout i, et donc chaque e_i est un k-échange et, finalement, μ est k-synchrone.

□

Comparaison avec [Bouajjani et al., 2018a] 1. *Les auteurs donnent une caractérisation pour les exécutions k-synchronisables dans [Bouajjani et al., 2018a, Section 5], celle-ci est similaire à la nôtre mais utilise le terme de "cycle" en lieu et place de "composante fortement connexe". Le papier dans sa globalité suggère qu'ils entendent par là un "cycle hamiltonien", c'est-à-dire un chemin cyclique qui ne passe pas deux fois par un même sommet. Cependant, l'absence de cycle hamiltonien de taille supérieure à k, et de label RS dans un cycle, n'assure pas la k-synchronisabilité du MSC.*

En effet, considérons l'Exemple 4.1.3 qui constitue un contre-exemple à cette caractérisation.

Exemple 4.1.3 – Le MSC de la Figure 4.4.a est 5-synchrone (et pas 4-synchrone). On constate que les 5 messages doivent être dans le même k-échange, il n'est pas possible d'en extraire un ou plusieurs afin de faire plusieurs k-échanges. Si l'on regarde le graphe de dépendances, en Figure 4.4.b, on constate que le plus grand cycle hamiltonien est de taille 4 tandis que la plus grande composante fortement connexe contient les 5 messages. Cet exemple constitue donc un contre-exemple pour la caractérisation graphique d'un MSC k-synchrone présente dans [Bouajjani et al., 2018a].

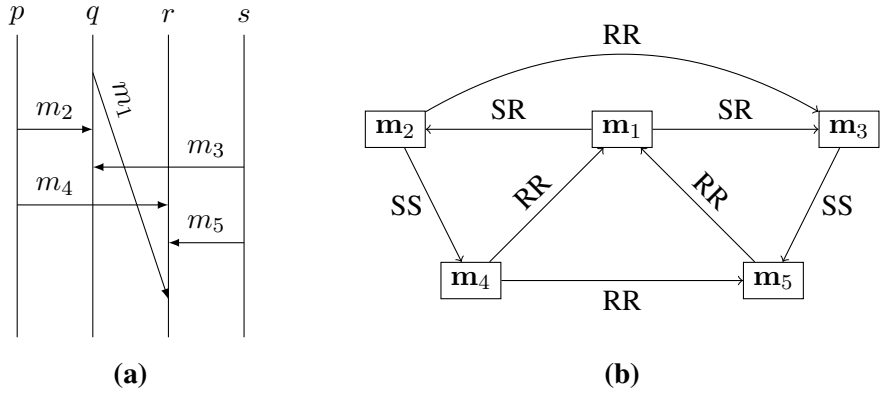


FIGURE 4.4 – Un MSC 5-synchrone (a) et son graphe de conflits (b)

Par conséquent, l'algorithme présenté dans [Bouajjani et al., 2018a] pour décider si un système est k-synchronisable n'est pas correct : le MSC de la Figure 4.4 est considéré à tord comme 4-synchrone selon cet algorithme, ce qui n'est pas le cas.

4.2 Décidabilité de l'accessibilité

On souhaite montrer que l'accessibilité d'un état de contrôle dans un système k-synchronisable est décidable. Comme le système étudié est k-synchronisable, toute exécution du système est k-synchronisable et donc divisible en k-échange.

Pour démontrer l'accessibilité, nous cherchons à constuire un système de transition qui contient dans ses configurations les états du système, et dans ses transitions, des k-échanges. Nous prouverons ensuite que ce système de transition est fini et régulier.

En effet, on peut recréer toutes les exécutions du système en concaténant des k-échanges. Pour concaténer deux k-échanges, il faut que l'état de contrôle d'arrivée du premier k-échange corresponde à l'état de contrôle de départ du second. Nous n'avons pas besoin de connaître le contenu exact des canaux car les messages présents ne seront pas lus dans les k-échanges suivants. Ceci est un avantage : se souvenir de tous les messages rendrait notre système de transition infini. Cependant, nous ne pouvons pas totalement ignorer le contenu des canaux.

Exemple 4.2.1 – Soit deux échanges e_1 et e_2 représentés par dans la Figure 4.5.a. Supposons que dans $\mathcal{S}$, $(\vec{\ell_0}, \vec{c_\emptyset}) \overset{e_1}{\Longmapsto} (\vec{\ell}, \vec{c})$ et $(\vec{\ell}, \vec{c_\emptyset}) \overset{e_2}{\Longmapsto} (\vec{\ell'}, \vec{c'})$. Ainsi, e_1 et e_2 sont des k-échanges : l'on peut organiser les envois suivis des réceptions, et ils existent dans $\mathcal{S}$ en respectant les contraintes

de la communication en boîte aux lettres. Cependant, le MSC $e_1 \cdot e_2$ n'est pas mb-réalisable. On constate en effet grâce au graphe de dépendances étendu associé de la Figure 4.5.b qu'il existe un cycle SS. Le message m_1 doit nécessairement être envoyé avant m_4. Pourtant, m_4 est couplé tandis que m_1 non, ce qui empêcherait donc m_4 d'être reçu.

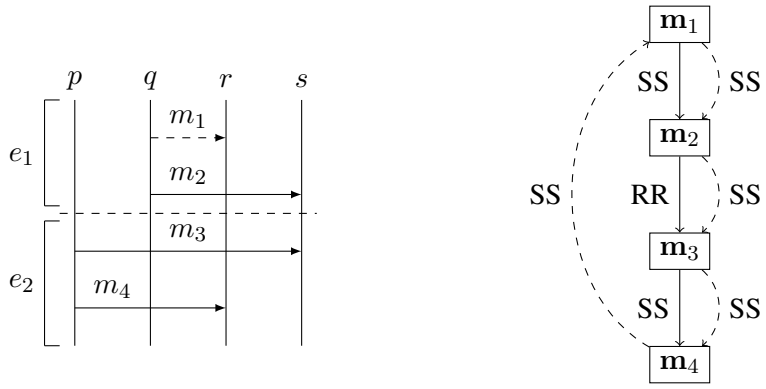


FIGURE 4.5 – Un MSC (a) et son graphe de dépendances étendu (b)

Ainsi, nous avons besoin de se souvenir d'assez d'informations des k-échanges précédents pour être capables de repérer un cycle SS comme défini dans le Théorème 4.1.1.

Tout d'abord, une observation cruciale est à faire : seuls les arcs générés par la RÈGLE 4 peuvent "remonter le temps". La procédure consiste donc à lire les k-échanges un par un, tout en cherchant des arcs produits par cette règle. Cela signifie que l'on doit garder assez d'informations des k-échanges précédents pour déterminer si un message est relié à un message non couplé d'un k-échange précédent. Il faut alors déterminer si l'arc qui les relie est contenu dans un cycle. Il est impossible de se souvenir de toutes les actions passées. Ainsi, les informations essentielles sont réduites à deux ensembles $\mathcal{C}_{\mathtt{S},p}$ et $\mathcal{C}_{\mathtt{R},p}$ qui collectent les informations suivantes : un processus q est dans $\mathcal{C}_{\mathtt{S},p}$ s'il effectue l'envoi d'un message causalement lié à un envoi à p non couplé qui le précède ; un processus q appartient à $\mathcal{C}_{\mathtt{R},p}$ s'il reçoit un message qui a été envoyé après un message non couplé à destination de p.

Plus précisément, nous avons :

$$\mathcal{C}_{\mathtt{S},p} = \{\mathsf{proc}_{\mathtt{S}}(\mathbf{m}) \mid \mathbf{m}' \overset{\text{SS}}{\dashrightarrow} \mathbf{m} \;\&\; \mathbf{m}' \text{ est non couplé} \;\&\; \mathsf{proc}_{\mathtt{R}}(\mathbf{m}') = p\}$$

$$\mathcal{C}_{\mathtt{R},p} = \{\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) \mid \mathbf{m}' \overset{\text{SS}}{\dashrightarrow} \mathbf{m} \;\&\; \mathbf{m}' \text{ est non couplé} \;\&\; \mathsf{proc}_{\mathtt{R}}(\mathbf{m}') = p \;\&\; \mathbf{m} \cap \mathtt{R} \neq \emptyset\}$$

Ces ensembles permettent de conserver seulement l'information nécessaire d'un k-échange à un autre afin de détecter ce qui empêcherait d'être mb-réalisable. Chaque k-échange est étudié à travers son graphe de dépendances étendu, que l'on désigne comme *local*, en opposition au graphe de dépendances étendu de la séquence de k-échanges, désigné comme *global*. Seuls les ensembles $\mathcal{C}_{\mathtt{S},p}$ et $\mathcal{C}_{\mathtt{R},p}$ seront transmis au k-échange suivant, qui lui-même y ajoutera les informations liées à ses propres actions, et ainsi de suite, jusqu'à la fin de la séquence.

Plus précisément, soit $e = s_1 \cdots s_m \cdot r_1 \cdots r_{m'}$ un k-échange, $\mathsf{GDE}(e) = (V, E)$ son graphe de dépendances étendu local, $\mathbb{B} = (2^{\mathbb{P}} \times 2^{\mathbb{P}})$ l'ensemble des abstractions de canaux et $\mathcal{B} : \mathbb{P} \to \mathbb{B}$

la fonction qui associe chaque $p \in \mathbb{P}$ les deux ensembles $\mathcal{B}(p) = (\mathcal{C}_{\mathtt{S},p}, \mathcal{C}_{\mathtt{R},p})$. Alors, le graphe de dépendances étendu $\mathsf{GDE}(e, \mathcal{B})$ est le graphe (V', E') avec $V' = V \cup \{\psi_p \mid p \in \mathbb{P}\}$ et $E' \supseteq E$ comme défini ci-après. Pour chaque processus $p \in \mathbb{P}$, le *sommet de synthèse* ψ_p résume tous les messages non couplés passés envoyés à p apparus dans les k-échanges précédents e. E' est l'ensemble E des arcs $\overset{\mathtt{XY}}{\dashrightarrow}$ du graphe de dépendances étendu sur les messages de e auxquels on a ajouté un ensemble d'arcs supplémentaires prenant en compte les sommets de synthèse définis ci-dessous :

$$\{\psi_p \xrightarrow{\mathtt{SX}} \mathbf{m} \mid \mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \in \mathcal{C}_{\mathtt{S},p} \;\&\; \mathbf{m} \cap \mathtt{X} \neq \emptyset, \mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}\} \tag{4.1}$$

$$\cup\{\psi_p \xrightarrow{\mathtt{SS}} \mathbf{m} \mid \mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \in \mathcal{C}_{\mathtt{R},p} \;\&\; \mathbf{m} \cap \mathtt{R} \neq \emptyset, \mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}\} \tag{4.2}$$

$$\cup\{\psi_p \xrightarrow{\mathtt{SS}} \mathbf{m} \mid \mathsf{proc}_{\mathtt{R}}(\mathbf{m}) \in \mathcal{C}_{\mathtt{R},p} \;\&\; \mathbf{m} \text{ est non couplé}\} \tag{4.3}$$

$$\cup\{\mathbf{m} \xrightarrow{\mathtt{SS}} \psi_p \mid \mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = p \;\&\; \mathbf{m} \cap \mathtt{R} \neq \emptyset\} \cup \{\psi_q \xrightarrow{\mathtt{SS}} \psi_p \mid p \in \mathcal{C}_{\mathtt{R},q}\} \tag{4.4}$$

Ces arcs supplémentaires résument les connexions vers et depuis les k-échanges précédents. Des exemples de k-échanges correspondants aux descriptions suivantes sont donnés Figures 4.6 et 4.7 avec l'arc créé en conséquence. Mais plus précisément :

- la Formule 4.1 crée un lien de ψ_p à $\mathbf{m}$ si l'envoi d'un message $\mathbf{m}'$, causalement lié à un message non couplé envoyé à p, est nécessaire et sur le même processus que l'action dans $\mathbf{m} \cap \mathtt{X}$ (voir Figure 4.6.a).
- la Formule 4.2 crée un lien de ψ_p à $\mathbf{m}$ si la réception d'un message $\mathbf{m}'$, causalement lié à un message non couplé envoyé à p, est nécessaire et sur le même processus qu'une action de $\mathbf{m}$ (voir Figure 4.6.b).
- la Formule 4.3 crée un lien entre un message non couplé et des messages couplés précédemment vus, causalement liés à un message non couplé envoyé à p (voir Figure 4.6.c).

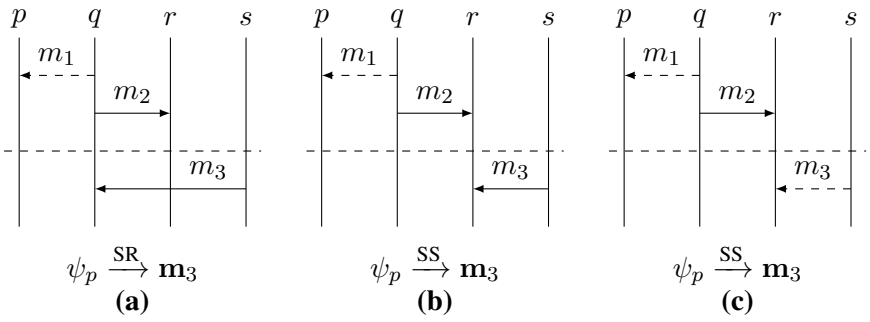


FIGURE 4.6 – Des exemples correspondants aux Formules 4.1 (a), 4.2 (b), et 4.3 (c)

- la Formule 4.4 crée un lien entre un message couplé envoyé à p et un message non couplé abstrait par ψ_p (voir Figures 4.7.a et 4.7.b).

Le graphe de dépendances étendu local est donc consititué des arcs du graphe de dépendances et des arcs des Formules 4.1 à 4.4 complétés par les arcs étendus définis dans la Figure 4.1.

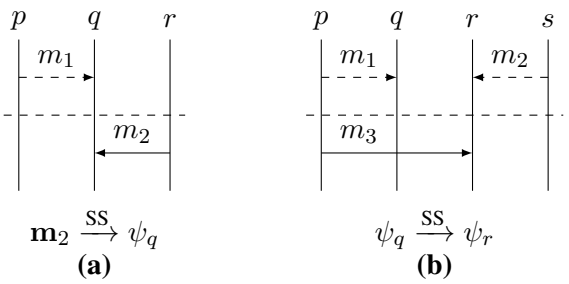


FIGURE 4.7 – Des exemples correspondants à la Formule 4.4 (a) et (b)

Finalement, posons $\mathcal{S}$ un système et $\xRightarrow[\text{real}]{e,k}$ la relation de transition donnée dans la Figure 4.8 sur des configurations abstraites de la forme $(\vec{\ell}, \mathcal{B})$: $\vec{\ell}$ est un état de contrôle global de $\mathcal{S}$ et $\mathcal{B} : \mathbb{P} \to \mathbb{B}$ est la fonction définie ci-dessus qui associe à chaque processus p une paire d'ensembles de processus $\mathcal{B}(p) = (\mathcal{C}_{\mathtt{S},p}, \mathcal{C}_{\mathtt{R},p})$. La transition $\xRightarrow[\text{real}]{e,k}$ met à jour ces ensembles en fonction du k-échange courant e. La séquence de k-échanges est mb-réalisable si pour tout $p \in \mathbb{P}$, $p \notin \mathcal{C}'_{\mathtt{R},p}$, ce qui signifie qu'il n'existe pas de dépendance cyclique comme décrite dans le Théorème 4.1.1. L'état initial est $(\vec{\ell_0}, \mathcal{B}_\emptyset)$ où $\mathcal{B}_\emptyset : \mathbb{P} \to \mathbb{B}$ décrit la fonction telle que $\mathcal{B}_\emptyset(p) = (\emptyset, \emptyset)$ pour tout $p \in \mathbb{P}$.

$$\frac{\begin{array}{c} e = s_1 \cdots s_m \cdot r_1 \cdots r_{m'} \in \mathtt{S}^m \cdot \mathtt{R}^{m'} \quad 0 \leq m' \leq m \leq k \\ (\vec{\ell}, \vec{c_\emptyset}) \overset{e}{\Rightarrow} (\vec{\ell'}, \vec{c}) \text{ pour un certain } \vec{c} \\ (\forall p \in \mathbb{P}) \quad \mathcal{B}(p) = (\mathcal{C}_{\mathtt{S},p}, \mathcal{C}_{\mathtt{R},p}), \quad \mathcal{B}'(p) = (\mathcal{C}'_{\mathtt{S},p}, \mathcal{C}'_{\mathtt{R},p}), \\ \mathsf{Unm}_p = \{\psi_p\} \cup \{\mathbf{m} \mid \mathbf{m} \text{ est non couplé}, \mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = p\}, \\ \mathcal{C}'_{\mathtt{X},p} = \mathcal{C}_{\mathtt{X},p} \cup \{p \mid p \in \mathcal{C}_{\mathtt{X},q}, \mathbf{m} \overset{\text{SS}}{\dashrightarrow} \psi_q, (\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = p \text{ ou } \mathbf{m} = \psi_p)\} \cup \\ \{\mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \mid \mathbf{m} \in \mathsf{Unm}_p \cap V, X = \mathtt{S}\} \cup \\ \{\mathsf{proc}_{\mathtt{X}}(\mathbf{m}') \mid \mathbf{m} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}', \mathbf{m} \in \mathsf{Unm}_p, \mathbf{m} \cap X \neq \emptyset\} \\ (\forall p \in \mathbb{P}) \quad p \notin \mathcal{C}'_{\mathtt{R},p} \end{array}}{(\vec{\ell}, \mathcal{B}) \xRightarrow[\text{real}]{e,k} (\vec{\ell'}, \mathcal{B}')}$$

FIGURE 4.8 – Définition de la relation de transition $\xRightarrow[\text{real}]{e,k}$

Poursuivons avec un exemple montrant les vues globale et locales du graphe de dépendances étendu du précédent MSC (de l'Exemple 4.2.1) et la construction des ensembles $\mathcal{C}_{\mathtt{S},p}$ et $\mathcal{C}_{\mathtt{R},p}$.

Exemple 4.2.2 – Dans cet exemple, nous construirons pas à pas les ensembles $\mathcal{C}_{\mathtt{S},r}$ et $\mathcal{C}_{\mathtt{R},r}$. Suivons alors la Figure 4.9.a pour le MSC, Figure 4.9.b pour les ensembles et Figure 4.9.c pour les graphes de dépendances étendus locaux. Le code couleur est le suivant : l'orange représente les ensembles $\mathcal{C}_{\mathtt{S},r}$, un processus devient orange lorsqu'il est ajouté à cet ensemble ; le vert représente les ensembles $\mathcal{C}_{\mathtt{R},r}$, un processus devient vert lorsqu'il est ajouté à cet ensemble.

Pour e_1, ces ensembles sont vides à l'initialisation. Nous pouvons ajouter q dans $\mathcal{C}_{\mathtt{S},r}$ car m_1 est non couplé et envoyé à r ($6^{ème}$ ligne de $\xRightarrow[\mathsf{real}]{e,k}$). Le message m_2 est envoyé par $q \in \mathcal{C}_{\mathtt{S},r}$ à s, ce qui nous permet d'ajouter s à $\mathcal{C}_{\mathtt{R},p}$ ($7^{ème}$ ligne de $\xRightarrow[\mathsf{real}]{e,k}$). Comme il s'agit du premier k-échange, les sommets de synthèse sont inutiles.

Regardons à présent le k-échange e_2. Nous ajoutons dans un premier temps le sommet ψ_r car $\mathcal{C}_{\mathtt{S},r} \neq \emptyset$. On voit alors dans le graphe de dépendances étendu de la Figure 4.9.c qu'on peut ajouter un arc $\psi_r \xrightarrow{\text{SS}} \mathbf{m}_3$ grâce à la Formule 4.2. Le graphe de dépendances étendu contient alors un arc $\psi_r \xdashrightarrow{\text{SS}} \mathbf{m}_4$ grâce à la RÈGLE 5. Les ensembles peuvent alors mettre être mis à jour. Comme $\psi_r \xrightarrow{\text{SS}} \mathbf{m}_3$, alors p qui envoie $\mathbf{m}_3$ est ajouté à $\mathcal{C}'_{\mathtt{S},r}$. Et comme $\psi_r \xdashrightarrow{\text{SS}} \mathbf{m}_4$, r qui reçoit $\mathbf{m}_4$ est ajouté à $\mathcal{C}'_{\mathtt{R},r}$.

Alors, on voit que $r \in \mathcal{C}'_{\mathtt{R},r}$ à la fin de e_2. Ainsi, cela signifie qu'un message reçu par r a un lien SS avec un message non couplé envoyé à r. Ce MSC $e_1 \cdot e_2$ ne peut donc pas être mb-réalisable.

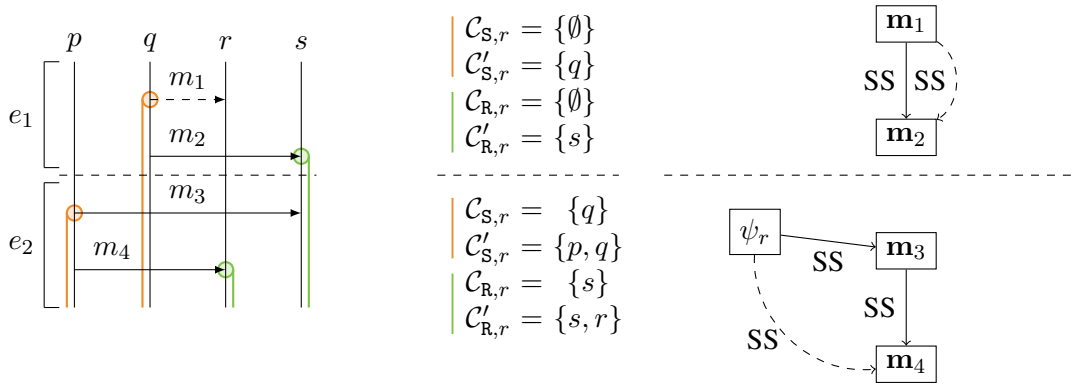


FIGURE 4.9 – Un MSC composé des k-échanges e_1 et e_2 avec l'évolution des ensembles $\mathcal{C}_{\mathtt{S},r}$ (en orange) et $\mathcal{C}_{\mathtt{R},r}$ (en vert) (a), le détail de l'évolution de ces ensembles pour chacun des k-échanges (b) et les graphes de dépendances locaux de ses k-échanges (c)

Comparaison avec [Bouajjani et al., 2018a] 2. *Une fonction de transition équivalente à* $\xRightarrow[\mathsf{real}]{e,k}$ *est définie dans [Bouajjani et al., 2018a, Figure 7]. Elle diffère par l'absence de caractérisation graphique explicite. En lieu et place, un ensemble $B(p)$ est calculé pour chaque processus p, il contient les processus qui ont envoyé un message non couplé à p, et ceux ayant reçu un message d'un processus appartenant déjà à l'ensemble. Le but est d'assurer que tout message suivant un message non couplé ne peut être couplé. Cette vérification est cohérente avec leur définition de délivrance causale mais n'est pas suffisante pour la suite de la preuve, d'où les changements effectués. Nous verrons dans les prochaines étapes les problèmes que cette version peut engendrer. Mais ce sont pour ces raisons que nous introduisons alors des ensembles $\mathcal{B}(p)$ plus élaborés, les graphe de dépendances étendu, la caractérisation graphique d'un MSC* mb*-réalisable et les sommets de synthèse.*

Lemme 4.2.1. *Un MSC μ est k-synchrone si et seulement s'il existe une linéarisation $e = e_1 \cdots e_n$ telle que $(\vec{\ell_0}, \mathcal{B}_\emptyset) \xrightarrow[\text{real}]{e_1,k} \cdots \xrightarrow[\text{real}]{e_n,k} (\vec{\ell}, \mathcal{B})$ pour un certain état global $\vec{\ell}$ et $\mathcal{B} : \mathbb{P} \to \mathbb{B}$.*

Remarque 4.2.1 – Il n'est pas nécessaire de vérifier des motifs qui ne sont pas mb-réalisables au sein même d'un k-échange, comme les cas de la Figure 4.10. En effet, ce ne sont pas des k-échanges puisqu'ils se sont pas réalisables, comme le précise la Définition 3.2.2.

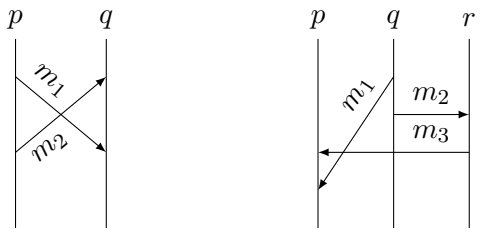


FIGURE 4.10 – Des MSC qui ne peuvent pas être des k-échanges car ils ne sont pas mb-réalisables

Il y a un nombre fini de configurations abstraites de la forme $(\vec{\ell}, \mathcal{B})$. De plus, comme $\mathbb{V}$ est fini, le nombre de k-échanges pour un k donné l'est aussi. Donc, $\xrightarrow[\text{real}]{e,k}$ est une relation sur un ensemble fini et l'ensemble $\text{MSC}_k(\mathcal{S})$ de MSC k-synchrones du système $\mathcal{S}$ forme un langage régulier. Ainsi, le problème suivant est décidable : une configuration abstraite $(\vec{\ell}, \mathcal{B})$ donnée est-elle accessible grâce à une exécution k-synchronisable depuis la configuration initiale ?

Théorème 4.2.2. *Soit $\mathcal{S}$ un système k-synchronisable et $\vec{\ell}$ un état de contrôle global de $\mathcal{S}$. Savoir si $\vec{\ell}$ est accessible est décidable.*

Démonstration.

Il existe un nombre fini de configurations abstraites de la forme $(\vec{\ell}, \mathcal{B})$ avec $\vec{\ell}$ un état de contrôle global de $\mathcal{S}$ et $\mathcal{B} : \mathbb{P} \to \mathbb{B}$. Alors $\xrightarrow[\text{real}]{e,k}$ est une relation sur un ensemble fini, et l'ensemble de MSC k-synchrone $\text{MSC}_k(\mathcal{S})$ du système $\mathcal{S}$ forme un langage régulier. Ainsi, il est décidable de savoir si une configuration abstraite donnée de la forme $(\vec{\ell}, \mathcal{B})$ est accessible depuis la configuration initiale suivant une exécution k-synchronisable étant une linéarisation d'un MSC contenu dans $\text{MSC}_k(\mathcal{S})$.

□

Remarque 4.2.2 – La régularité de l'ensemble des MSC k-synchronisables implique la décidabilité d'autres propriétés pour un système k-synchronisable telles que l'absence d'interblocages, de messages inattendus ou de messages non couplé.

4.3 Décidabilité de la k-synchronisabilité

Dans cette section, nous établirons la décidabilité de la k-synchronisabilité pour un k donné. Notre approche est similaire à celle présentée dans [Bouajjani et al., 2018a] et basée sur la notion

d'exécution critique (*borderline violation* en anglais), notion que nous présenterons dans la première partie de cette section. Cependant, nous procédons à des ajustements selon la correction de la caractérisation d'une exécution k-synchronisable (Théorème 4.1.2).

4.3.1 Description de la démarche

La preuve s'appuie sur une observation primordiale : si un système n'est pas k-synchronisable, cela implique l'existence d'une exécution *k-critique*. La preuve de décidabilité de la k-synchronisabilité consiste donc en la recherche d'une telle exécution.

Définition 4.3.1 (Exécution k-critique)**.** Une exécution e est k-critique si et seulement si $e = e' \cdot r$, avec r une réception n'est pas k-synchronisable et e' une exécution k-synchronisable.

Un système $\mathcal{S}$ qui n'est pas k-synchronisable admet toujours au moins une exécution k-critique $e' = e \cdot r \in \text{Ex}(\mathcal{S})$ avec $r \in \mathtt{R}$, où e est un unique préfixe minimal k-synchronisable. Puisque e est k-synchronisable contrairement à e', r ne peut être un k-échange à lui tout seul, et c'est pourquoi il s'agit toujours d'une réception. Pour trouver une telle exécution, Bouajjani *et al.* ont introduit un système instrumenté $\mathcal{S}'$ [Bouajjani et al., 2018a, Section 6.2] qui se comporte comme $\mathcal{S}$, à l'exception qu'il contient un processus supplémentaire π et qu'un, et un seul, message non déterministiquement choisi peut lui être envoyé. Ce message sera, à la fin de l'exécution, renvoyé au destinataire original, retrouvé car ajouté dans l'étiquette du message. Ainsi, pour chaque réception possible, chaque processus à la possibilité de recevoir le message en question en provenance du processus π au lieu de l'expéditeur initial. Depuis son état initial, le processus π peut recevoir n'importe quel message depuis le système. Chaque réception l'amène dans un état différent, depuis lequel il sera capable de transférer le message au destinataire original. Une fois le message transféré, π atteint un état final et reste alors inactif.

Définition 4.3.2 (Système instrumenté)**.** Soit $\mathcal{S} = ((L_p, \delta_p, \ell_p^0) \mid p \in \mathbb{P})$ un système d'automates communicants. Le système instrumenté $\mathcal{S}'$ associé à $\mathcal{S}$ est défini tel que $\mathcal{S}' = \Big((L_p, \delta'_p, \ell_p^0) \mid p \in \mathbb{P} \cup \{\pi\}\Big)$ où pour chaque $p \in \mathbb{P}$:

$$\begin{aligned}\delta'_p = \delta_p \cup \{\ell \xrightarrow{\mathtt{s}(p,\pi,(q,m))}_p \ell' \mid \ell \xrightarrow{\mathtt{s}(p,q,m)}_p \ell' \in \delta_p\} \\ \cup \{\ell \xrightarrow{\mathtt{r}(\pi,p,m)}_p \ell' \mid \ell \xrightarrow{\mathtt{r}(q,p,m)}_p \ell' \in \delta_p\}\end{aligned}$$

Le processus π est l'automate communicant $(L_\pi, \ell_\pi^0, \delta_\pi)$ où

- $L_\pi = \{\ell_\pi^0, \ell_f\} \cup \{\ell_{(q,m)} \mid m \in \mathbb{V}, q \in \mathbb{P}\}$, et
- $\delta_\pi = \{\ell_\pi^0 \xrightarrow{\mathtt{r}(p,\pi,(q,m))}_\pi \ell_{(q,m)} \mid \mathtt{s}(p,q,m) \in \mathtt{S}\}$
  $\cup \{\ell_{(q,m)} \xrightarrow{\mathtt{s}(\pi,q,m)}_\pi \ell_f \mid \mathtt{r}(p,q,m) \in \mathtt{R}\}$

L'exemple qui suit illustre le comportement d'un système instrumenté.

Exemple 4.3.1 – Soit e_1 une exécution d'un système $\mathcal{S}$ correspondant au MSC μ_1 de la Figure 4.11.a. e_1 n'est pas 1-synchronisable et elle est 1-critique dans $\mathcal{S}$. Si l'on supprime la dernière réception, $\mathtt{r}(p,q,m_1)$, elle devient en effet 1-synchronisable. Soit μ_2 de la Figure 4.11.b le MSC obtenu par la déviation du message m_1 par le processus π dans le système $\mathcal{S}'$, le message m_1 est ensuite renvoyé au processus q.

On voit alors que le MSC μ_2 est 1-synchrone. Dans ce cas, le système instrumenté $\mathcal{S}'$ dans la sémantique 1-synchrone contient une exécution 1-critique de $\mathcal{S}$.

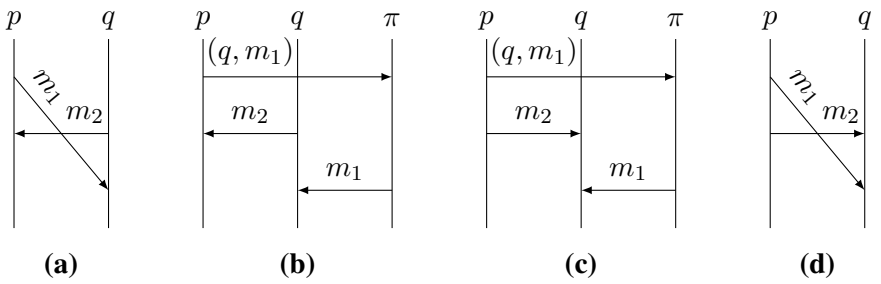


FIGURE 4.11 – Les MSC μ_1 (a), μ_2 (b), μ_3 (c) et μ_4 (d)

Pour toute exécution $e \cdot r \in \text{EX}(\mathcal{S})$, qui termine donc par une réception, il existe une exécution $\mathsf{deviate}(e \cdot r) \in \text{EX}(\mathcal{S}')$ où le message associé à cette réception r est dévié par π. Formellement, si $e \cdot r = e_1 \cdot s \cdot e_2 \cdot r$ avec $r = \mathtt{r}(p, q, m)$ et $s \mapsto r$ alors

$$\mathsf{deviate}(e \cdot r) = e_1 \cdot \mathtt{s}(p, \pi, (q, m)) \cdot \mathtt{r}(p, \pi, (q, m)) \cdot e_2 \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$$

On s'intéresse alors à deux nouveaux types d'exécutions : les exécutions faisables et les mauvaises exécutions. La première décrit une exécution du système instrumenté qui correspond effectivement à une exécution dans le système original, modulo la déviation d'un message. La seconde décrit alors une exécution faisable et k-critique dans le système original.

Définition 4.3.3 (Exécution faisable, mauvaise exécution)**.** Une exécution k-synchronisable e' du système $\mathcal{S}'$ est faisable s'il existe une exécution $e \cdot r \in \text{EX}(\mathcal{S})$ telle que $\mathsf{deviate}(e \cdot r) = e'$. Une exécution faisable $e' = \mathsf{deviate}(e \cdot r)$ de $\mathcal{S}'$ est mauvaise si l'exécution $e \cdot r$ n'est pas k-synchronisable dans $\mathcal{S}$.

Exemple 4.3.2 – L'exécution e_2 représentée par son MSC dans la Figure 4.11.b est 1-synchrone dans $\mathcal{S}'$. Elle est la déviation de l'exécution e_1 de la Figure 4.11.a qui est mb-réalisable. Ainsi, e_2 est une exécution faisable. Cependant, e_1 n'est pas 1-synchrone et donc e_2 est une mauvaise exécution.

Soit e_3 une exécution, dont on peut voir son MSC $\mu_3 = msc(e_3)$ dans la Figure 4.11.c. Cet MSC est visiblement 1-synchrone et est mb-réalisable dans $\mathcal{S}'$. Cependant, la séquence d'action e_4, MSC en Figure 4.11.d, telle que $\mathsf{deviate}(e_4) = e_3$, n'est pas mb-réalisable et donc ne correspond à aucune exécution du système original $\mathcal{S}$. En d'autres mots, l'exécution e_3 n'est pas faisable.

La recherche d'une exécution faisable mais mauvaise permet alors de déduire si le système étudié est k-synchronisable.

Lemme 4.3.1. *Un système $\mathcal{S}$ n'est pas k-synchronisable si et seulement s'il existe une exécution e' de $\mathcal{S}'$ qui est faisable et mauvaise.*

Démonstration.

$\Rightarrow$ Soit $\mathcal{S}$ un système non k-synchronisable. Alors il existe une exécution qui n'est pas k-synchronisable, qui contient un unique préfixe minimal de la forme $e \cdot r$ avec e une exécution k-synchronisable et une réception $r = \mathtt{r}(p, q, m)$. Alors, e est mauvaise et il existe une exécution $e' = \mathsf{deviate}(e \cdot r) \in \mathrm{Ex}(\mathcal{S}')$.

Puisque e est k-synchronisable, $msc(e)$ est k-synchrone et il existe une linéarisation e'' telle que $e'' = e_1 \cdots e_n$ et il existe un k-échange e_i contenant une action d'envoi $\mathtt{s}(p, q, m)$. Maintenant, notons $e''' = e_1 \cdots e'_i \cdots e_n$ où le k-échange e'_i remplace le k-échange e_i. Dans e'_i, l'action $\mathtt{s}(p, q, m)$ est remplacée par $\mathtt{s}(p, \pi, (q, m))$ et $\mathtt{r}(p, q, m)$ par $\mathtt{r}(p, \pi, (q, m))$. L'exécution e''' est k-synchronisable dans $\mathrm{Ex}(\mathcal{S}')$. Finalement, $e''' \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$ est également k-synchronisable, les deux dernières actions formant un nouveau k-échange. Ainsi, e' est faisable.

$\Leftarrow$ S'il existe une exécution e' de $\mathcal{S}'$ qui est faisable et mauvaise. Alors, par construction, $e' = \mathsf{deviate}(e \cdot r)$ et $e \cdot r$ n'est pas k-synchronisable. Et donc, $\mathcal{S}$ n'est pas k-synchronisable, ce qui conclut cette preuve.

□

Comme nous l'avons déjà relevé, l'ensemble des MSC k-synchrones de $\mathcal{S}'$ est régulier. La procédure de décision de la k-synchronisabilité s'appuie sur le fait que l'ensemble des MSC qui ont une linéarisation faisable et mauvaise est lui aussi régulier. Un automate non-déterministe reconnaissant ce langage peut alors être construit. La décidabilité découle alors du Lemme 4.3.1 et de la décision du vide pour un automate fini non déterministe.

Plus exactement, les étapes seront les suivantes : il faudra dans un premier temps être capable de reconnaître les exécutions faisables parmi les exécutions du système instrumenté, et ensuite, parmi celles-ci, déterminer s'il existe une exécution mauvaise, et donc une exécution k-critique dans le système d'origine.

4.3.2 Reconnaissance des exécutions faisables

Nous commençons donc par la reconnaissance des exécutions faisables dans le système instrumenté. Pour cela, nous construisons un automate qui reconnaît de telles exécutions en partant de la construction précédente qui reconnaissait les séquences de k-échanges mb-réalisables.

Dans la suite, on suppose une exécution $e' \in \mathrm{Ex}(\mathcal{S}')$ qui contient exactement un envoi de la forme $\mathtt{s}(p, \pi, (q, m))$ et une réception de la forme $\mathtt{r}(\pi, q, m)$, cette réception étant la dernière action de e'. Soit $(V, \{\xrightarrow{\mathrm{XY}}_{X,Y \in \{S,R\}}\})$ le graphe de dépendances de e'. Il existe alors deux sommets $\mathbf{m}_{\mathsf{start}}, \mathbf{m}_{\mathsf{stop}} \in V$ déterminés de façon unique tels que $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}_{\mathsf{start}}) = \pi$ et $\mathsf{proc}_{\mathtt{S}}(\mathbf{m}_{\mathsf{stop}}) = \pi$, qui correspondent respectivement au premier et dernier messages de la déviation. Le graphe de dépendances de $e \cdot r$ est obtenu en regroupant ces deux sommets.

Exemple 4.3.3 – La Figure 4.12.a représente le graphe de dépendances $\mathsf{GD}(e_3)$, avec $msc(e_3)$ décrit dans la Figure 4.11.c. La Figure 4.12.b représente le graphe de dépendances $\mathsf{GD}(e_4)$, avec $msc(e_4)$ décrit dans la Figure 4.11.d. On constate que si l'on prend le premier et que l'on fusionne les sommets $\mathbf{m}_{\mathsf{start}} = (q, \mathbf{m}_1)$ et $\mathbf{m}_{\mathsf{stop}} = \mathbf{m}_1$, on obtient bien le second graphe. En l'occurrence, on remarque que le graphe de dépendances étendu $\mathsf{GDE}(e_4)$ contiendrait un cycle SS. Ainsi, e_4 n'est pas mb-réalisable et donc e_3 n'est pas faisable.

Pour vérifier qu'une exécution de $\mathcal{S}'$ est faisable, il faut alors chercher un chemin SS entre $\mathbf{m}_{\mathsf{start}}$ et $\mathbf{m}_{\mathsf{stop}}$, jouant le rôle d'un cycle dans l'exécution équivalente dans $\mathcal{S}$.

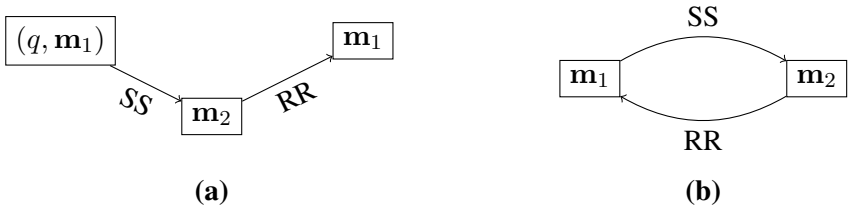


FIGURE 4.12 – Les graphes de dépendances $\mathsf{GD}(e_3)$ (a) et $\mathsf{GD}(e_4)$ (b)

Lemme 4.3.2. *Une exécution e' n'est pas faisable si et seulement s'il existe un sommet $\mathbf{m}$ dans le graphe de dépendances étendu $\mathsf{GDE}(e')$ tel que $\mathbf{m}_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m} \xrightarrow{\text{RR}} \mathbf{m}_{\mathsf{stop}}$.*

Démonstration.

$\Leftarrow$ S'il y a $\mathbf{m}$ tel que $\mathbf{m}_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m} \xrightarrow{\text{RR}} \mathbf{m}_{\mathsf{stop}}$, cela signifie qu'un message envoyé après un message dévié est reçu avant ce dernier : $msc(e')$ n'est pas mb-réalisable.

$\Rightarrow$ Supposons maintenant que e' n'est pas faisable. Cela implique que e est k-synchronisable et $e \cdot r$ n'est pas mb-réalisable. Ainsi, il existe un message non couplé dont l'action d'envoi est l'action a_i et un message couplé dans e dont l'envoi est l'action a_j, et, d'après la Définition 2.2.5, il existe i', j' tels que $r = a_{i'}$, $a_i \vdash a_{i'}$, $a_j \vdash a_{j'}$, et $i \prec j$ et $j' \prec i'$. Donc le graphe de dépendances $\mathsf{GD}(e \cdot r)$ contient deux sommets $\mathbf{m}_d = \{a_i, a_{i'}\}$ et $\mathbf{m} = \{a_j, a_{j'}\}$ tel que $\mathbf{m}_d \overset{\text{SS}}{\dashrightarrow} \mathbf{m} \xrightarrow{\text{RR}} \mathbf{m}_d$. À cause de la déviation, le sommet $\mathbf{m}_d$ est divisé en deux sommets $\mathbf{m}_{\mathsf{start}}$ et $\mathbf{m}_{\mathsf{stop}}$ dans $\mathsf{GD}(e')$, et donc nous pouvons conclure que $\mathbf{m}_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m} \xrightarrow{\text{RR}} \mathbf{m}_{\mathsf{stop}}$.

□

On veut alors décider si une exécution e' est faisable. Cela signifie qu'il faut déterminer s'il existe une action d'envoi $\mathtt{s}(p', q, m')$, faite causalement après $\mathbf{m}_{\mathsf{start}}$ couplée à une réception $\mathtt{r}(p', q, m')$, arrivant causalement avant la réception $\mathbf{m}_{\mathsf{stop}}$. Cela revient à traiter l'action d'envoi à π comme un message non couplé. Nous considérons donc les ensembles $\mathcal{C}^{\pi}_{\mathtt{S}}$ et $\mathcal{C}^{\pi}_{\mathtt{R}}$, similaires à ceux utilisés pour $\xrightarrow[\mathsf{real}]{e,k}$, mais dans le but de calculer quelles actions ont lieu causalement après l'envoi à π.

On introduit alors aussi un sommet de synthèse ψ_{start} et les arcs supplémentaires suivants les mêmes principes que dans la section précédente. Formellement, pour $\mathcal{B} : \mathbb{P} \to \mathbb{B}$, $\mathcal{C}^{\pi}_{\mathtt{S}}, \mathcal{C}^{\pi}_{\mathtt{R}} \subseteq \mathbb{P}$ et $e \in \mathtt{S}^{\leq k} \cdot \mathtt{R}^{\leq k}$ fixés, et $\mathsf{GD}(e, \mathcal{B}) = (V', E')$ le graphe de dépendances avec les sommets de synthèse pour les messages non couplés comme défini précédemment. Le graphe de dépendances local $\mathsf{GD}(e, \mathcal{B}, \mathcal{C}^{\pi}_{\mathtt{S}}, \mathcal{C}^{\pi}_{\mathtt{R}})$ est défini comme le graphe (V'', E'') où $V'' = V' \cup \{\psi_{\mathsf{start}}\}$ et E'' est l'ensemble E' auquel on ajoute les arcs suivants :

$$
\begin{aligned}
&\{\psi_{\mathsf{start}} \xrightarrow{\text{SX}} \mathbf{m} \mid \mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \in \mathcal{C}^{\pi}_{\mathtt{S}} \ \&\ \mathbf{m} \cap \mathtt{X} \neq \emptyset, \mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}\} \\
\cup\ &\{\psi_{\mathsf{start}} \xrightarrow{\text{SS}} \mathbf{m} \mid \mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \in \mathcal{C}^{\pi}_{\mathtt{R}} \ \&\ \mathbf{m} \cap \mathtt{R} \neq \emptyset, \mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}\} \\
\cup\ &\{\psi_{\mathsf{start}} \xrightarrow{\text{SS}} \mathbf{m} \mid \mathsf{proc}_{\mathtt{R}}(\mathbf{m}) \in \mathcal{C}^{\pi}_{\mathtt{R}} \ \&\ \mathbf{m} \text{ est non couplé } \} \ \cup\ \{\psi_{\mathsf{start}} \xrightarrow{\text{SS}} \psi_p \mid p \in \mathcal{C}^{\pi}_{\mathtt{R}}\}
\end{aligned}
$$

$$(\vec{\ell}, \mathcal{B}) \xRightarrow[\mathsf{real}]{e,k} (\vec{\ell'}, \mathcal{B}) \qquad e = a_1 \cdots a_n \qquad (\forall \mathbf{m})\ \mathsf{proc}_{\mathtt{S}}(\mathbf{m}) \neq \pi$$
$$(\forall \mathbf{m}, \mathbf{m}')\ \mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \mathsf{proc}_{\mathtt{R}}(\mathbf{m}') = \pi \implies \mathbf{m} = \mathbf{m}' \wedge \mathsf{dest}_\pi = \bot$$
$$(\forall \mathbf{m})\ \mathbf{m} \ni \mathtt{s}(p, \pi, (q, m)) \implies \mathsf{dest}'_\pi = q \quad \mathsf{dest}_\pi \neq \bot \implies \mathsf{dest}'_\pi = \mathsf{dest}_\pi$$
$$\mathcal{C}^{\pi\prime}_{\mathtt{X}} = \mathcal{C}^{\pi}_{\mathtt{X}} \cup \{\mathsf{proc}_{\mathtt{X}}(\mathbf{m}') \mid \mathbf{m} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}' \ \&\ \mathbf{m}' \cap \mathtt{X} \neq \emptyset \ \&\ (\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \pi \text{ ou } \mathbf{m} = \psi_{\mathsf{start}})\}$$
$$\cup \{\mathsf{proc}_{\mathtt{S}}(\mathbf{m}) \mid \mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \pi \ \&\ \mathtt{X} = \mathtt{S}\}$$
$$\cup \{p \mid p \in \mathcal{C}_{\mathtt{X},q} \ \&\ \mathbf{m} \overset{\mathrm{SS}}{\dashrightarrow} \psi_q \ \&\ (\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \pi \text{ ou } \mathbf{m} = \psi_{\mathsf{start}})\}$$
$$\mathsf{dest}'_\pi \notin \mathcal{C}^{\pi\prime}_{\mathtt{R}}$$

$$(\vec{\ell}, \mathcal{B}, (\mathcal{C}^{\pi}_{\mathtt{S}}, \mathcal{C}^{\pi}_{\mathtt{R}}), \mathsf{dest}_\pi) \xRightarrow[\mathsf{feas}]{e,k} (\vec{\ell}', \mathcal{B}', (\mathcal{C}^{\pi\prime}_{\mathtt{S}}, \mathcal{C}^{\pi\prime}_{\mathtt{R}}), \mathsf{dest}'_\pi)$$

FIGURE 4.13 – Définition de la relation de transition $\xRightarrow[\mathsf{feas}]{e,k}$

Comme tout à l'heure, on considère la "fermeture" de ces arcs définie par les règles de la Figure 4.1. La transition $\xRightarrow[\mathsf{feas}]{e,k}$, permettant la construction de ces ensembles, est définie dans la Figure 4.13.

Cette relation relie des configurations abstraites de la forme $(\vec{\ell}, \mathcal{B}, \mathcal{B}_\pi, \mathsf{dest}_\pi)$ où $\mathcal{B}_\pi = (\mathcal{C}^{\pi}_{\mathtt{S}}, \mathcal{C}^{\pi}_{\mathtt{R}})$ et $\mathsf{dest}_\pi \in \mathbb{P} \cup \{\bot\}$ garde en mémoire le destinataire original du message dévié. La configuration abstraite initiale est ici $(\vec{\ell_0}, \mathcal{B}_\emptyset, (\emptyset, \emptyset), \bot)$ où $\bot$ signifie que le destinataire n'a pas encore été identifié, le choix du message dévié n'étant pas encore fait. La valeur de dest_π sera alors mise à jour une fois le message envoyé à π.

Lemme 4.3.3. *Soit e' une exécution de $\mathcal{S}'$. Alors, e' est une exécution k-synchronisable et faisable si et seulement s'il existe $e'' = e_1 \cdots e_n \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$ telle que*

$$(\vec{\ell_0}, \mathcal{B}_\emptyset, (\emptyset, \emptyset), \bot) \xRightarrow[\mathsf{feas}]{e_1,k} \cdots \xRightarrow[\mathsf{feas}]{e_n,k} (\vec{\ell}', \mathcal{B}', \mathcal{B}'_\pi, q)$$

avec $e_1 \cdots e_n \in \mathtt{S}^{\leq k} \cdot \mathtt{R}^{\leq k}$, $\mathcal{B}' : \mathbb{P} \to \mathbb{B}$, $\mathcal{B}'_\pi \in 2^{\mathbb{P}}$ et un état global $\vec{\ell}'$ tels que $msc(e') = msc(e'')$, $q \notin \mathcal{C}'_{\mathtt{R},q}$, avec $\mathcal{B}'(q) = (\mathcal{C}'_{\mathtt{S},q}, \mathcal{C}'_{\mathtt{R},q})$.

Démonstration.

Posons tout d'abord quelques propriétés sur les variables $\mathcal{B}, \mathcal{B}_\pi$ et dest_π.

Soit e' une exécution k-synchronisable du système $\mathcal{S}'$ et $e'' = e_1 \cdots e_n$ telle que $msc(e') = msc(e'')$, et supposons qu'il existe $\mathcal{B}, \mathcal{B}_\pi, \mathsf{dest}_\pi$ tels que

$$(\vec{\ell_0}, \mathcal{B}_\emptyset, (\emptyset, \emptyset), \bot) \xRightarrow[\mathsf{feas}]{e_1,k} \ldots \xRightarrow[\mathsf{feas}]{e_n,k} (\vec{\ell}, \mathcal{B}, (\mathcal{C}^{\pi}_{\mathtt{S}}, \mathcal{C}^{\pi}_{\mathtt{R}}), \mathsf{dest}_\pi).$$

Notons que $\mathsf{GD}(e') = \mathsf{GD}(e'')$. Par récurrence sur n, on veut établir que

1. $\mathsf{dest}_\pi = q$ si et seulement si un message de la forme (q, m) a été envoyé à π dans e' ;
2. il y a au plus un message envoyé à π dans e' ;
3. soit $\mathbf{m}_{\mathsf{start}}$ le seul sommet dans $\mathsf{GD}(e')$ (s'il existe) tel que $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}_{\mathsf{start}}) = \pi$; pour tout $\mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}$:

$$C^{\pi}_{\mathtt{X}} = \{\mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \mid (\mathbf{m} \cap X \neq \emptyset \ \&\ \mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m} \text{ dans } \mathsf{GD}(e')) \text{ ou } (\mathbf{m}, \mathtt{X}) = (\mathbf{m}_{\mathsf{start}}, \mathtt{S})\}.$$

Les deux premiers points découlent de la définition de la relation $\xrightarrow[\mathsf{feas}]{e,k}$. Passons au dernier point. Le cas $n = 1$ est immédiat. Supposons alors que

$$(\vec{\ell_0}, \mathcal{B}_\emptyset, (\emptyset,\emptyset), \bot) \xrightarrow[\mathsf{feas}]{e_1,k} \dots \xrightarrow[\mathsf{feas}]{e_{n-1},k} (\vec{\ell}, \mathcal{B}, \mathcal{B}_\pi, \mathsf{dest}_\pi) \xrightarrow[\mathsf{feas}]{e_n,k} (\vec{\ell}', \mathcal{B}', \mathcal{B}'_\pi, \mathsf{dest}'_\pi)$$

avec

$$C^\pi_{\mathtt{X}} = \{\mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \mid (\mathbf{m} \cap \mathtt{X} \neq \emptyset \ \&\ \mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m} \text{ dans } \mathsf{GD}(e_1 \cdots e_{n-1})) \text{ ou } (\mathbf{m}, \mathtt{X}) = (\mathbf{m}_{\mathsf{start}}, \mathtt{S})\}$$

et donc, montrons que

$$C^{\pi\prime}_{\mathtt{X}} = \{\mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \mid (\mathbf{m} \cap \mathtt{X} \neq \emptyset \ \&\ \mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m} \text{ dans } \mathsf{GD}(e_1 \cdots e_n)) \text{ ou } (\mathbf{m}, \mathtt{X}) = (\mathbf{m}_{\mathsf{start}}, \mathtt{S})\}$$

- Soit $\mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}$ et $p \in C^{\pi\prime}_{\mathtt{X}}$ et montrons qu'il existe un sommet $\mathbf{m}$ tel que $p = \mathsf{proc}_{\mathtt{X}}(\mathbf{m})$ et soit $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SX}}{\dashrightarrow} \mathbf{m}$ dans $\mathsf{GD}(e_1 \cdots e_n)$ ou $(\mathbf{m}, \mathtt{X}) = (\mathbf{m}_{\mathsf{start}}, \mathtt{S})$. Nous raisonnons par une analyse de cas sur la raison pour laquelle $p \in C^{\pi\prime}_{\mathtt{X}}$, en partant de la définition de $C^{\pi\prime}_{\mathtt{X}}$ de la Figure 4.13.
  - $p \in C^\pi_{\mathtt{X}}$. Alors, d'après l'hypothèse de récurrence, il existe $\mathbf{m}$ tel que $p = \mathsf{proc}_{\mathtt{X}}(\mathbf{m})$, et $\mathbf{m}_{\mathsf{start}} \xrightarrow{\mathrm{SS}} \mathbf{m}$ dans $\mathsf{GD}(e_1 \cdots e_{n-1})$, et donc aussi dans $\mathsf{GD}(e_1 \cdots e_n)$, ou $(\mathbf{m}, \mathtt{X}) = (\mathbf{m}_{\mathsf{start}}, \mathtt{S})$.
  - $p = \mathsf{proc}_{\mathtt{X}}(\mathbf{m}')$, $\mathbf{m} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}'$, $\mathbf{m}' \cap X \neq \emptyset$, et $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \pi$, pour des messages $\mathbf{m}, \mathbf{m}'$ de e_n. Puisque $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \pi$, $\mathbf{m} = \mathbf{m}_{\mathsf{start}}$. Ce qui conclut ce cas.
  - $p = \mathsf{proc}_{\mathtt{X}}(\mathbf{m}')$, $\psi_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}'$, $\mathbf{m}' \cap X \neq \emptyset$, pour un message $\mathbf{m}'$ de e_n. Il reste à prouver que $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}'$. Comme $\psi_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}'$, il existe $\mathbf{m}, \mathtt{Y}$ tels que $\psi_{\mathsf{start}} \xrightarrow{\mathrm{SY}} \mathbf{m}$ dans $\mathsf{GD}(e_n, \mathcal{B}, \mathcal{B}_\pi)$, $\mathbf{m} \cap \mathtt{Y} \neq \emptyset$ et soit $\mathbf{m} \overset{\mathrm{YS}}{\dashrightarrow} \mathbf{m}'$, soit $(\mathbf{m}, \mathtt{Y}) = (\mathbf{m}', \mathtt{S})$. Nous raisonnons par analyse de cas sur la construction de cet arc $\psi_{\mathsf{start}} \xrightarrow{\mathrm{SY}} \mathbf{m}$.
    - $\mathsf{proc}_{\mathtt{Y}}(\mathbf{m}) \in \mathcal{C}^\pi_{\mathtt{S}}$ et $\mathbf{m} \cap Y \neq \emptyset$. Soit $q = \mathsf{proc}_{\mathtt{Y}}(\mathbf{m})$. Puisque $q \in \mathcal{C}^\pi_{\mathtt{S}}$, par hypothèse de récurrence il existe $\mathbf{m}_1$ dans un k-échange précédent tel que $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}_1$ dans $\mathsf{GD}(e_1 \cdots e_{n-1})$ ou $\mathbf{m}_1 = \mathbf{m}_{\mathsf{start}}$. Puisque $\mathsf{proc}_{\mathtt{S}}(\mathbf{m}_1) = \mathsf{proc}_{\mathtt{Y}}(\mathbf{m})$, il existe un arc $\mathbf{m}_1 \xrightarrow{\mathrm{SY}} \mathbf{m}$ dans $\mathsf{GD}(e_1 \cdots e_n)$. Par hypothèse de récurrence, on sait aussi que soit $\mathbf{m} \overset{\mathrm{YS}}{\dashrightarrow} \mathbf{m}'$ soit $(\mathbf{m}, Y) = (\mathbf{m}', \mathtt{S})$. Dans les deux cas, nous avons $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}_1 \xrightarrow{\mathrm{SS}} \mathbf{m}'$, ou $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}'$ quand $\mathbf{m}_1 = \mathbf{m}_{\mathsf{start}}$.
    - $\psi_{\mathsf{start}} \xrightarrow{\mathrm{SS}} \mathbf{m}$, $\mathsf{proc}_{\mathtt{Y}}(\mathbf{m}) \in \mathcal{C}^\pi_{\mathtt{R}}$ et $\mathbf{m} \cap \mathtt{R} \neq \emptyset$. Encore par hypothèse, on a $\mathbf{m}_1$ tel que $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}_1 \xrightarrow{\mathrm{RY}} \mathbf{m}$, et donc $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}'$.
    - $\psi_{\mathsf{start}} \xrightarrow{\mathrm{SS}} \mathbf{m}$, $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) \in \mathcal{C}^\pi_{\mathtt{R}}$ et $\mathbf{m}$ non couplé. Encore par hypothèse, nous avons $\mathbf{m}_1$ tel que $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}_1 \xrightarrow{\mathrm{RS}} \mathbf{m}$. Si $\mathbf{m} = \mathbf{m}'$, nous avons $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}'$, ce qui conclut ce cas. Sinon, comme $\mathbf{m} \overset{\mathrm{YS}}{\dashrightarrow} \mathbf{m}'$ et $\mathbf{m}$ non couplé, on peut déduire $\mathbf{m} \xrightarrow{\mathrm{SS}} \mathbf{m}'$; enfin, on a $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}_1 \xrightarrow{\mathrm{RS}} \mathbf{m} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}'$, donc $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}'$, ce qui conclut ce cas.
    - $v = \psi_q$ pour un certain $q \in \mathcal{C}^\pi_{\mathtt{R}}$. Puisque ψ_q n'a pas d'arc sortant étiquetés RS, $\psi_q \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}'$. Comme $q \in \mathcal{C}^\pi_{\mathtt{R}}$, on obtient par hypothèse de récurrence qu'il existe

un sommet $\mathbf{m}_1$ tel que $\mathbf{m}_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_1$ et $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}_1) = q$. Comme nous l'avons vu dans la preuve du Lemme 4.2.1, $\psi_q \overset{\text{SS}}{\dashrightarrow} \mathbf{m}'$ implique qu'il existe un sommet $\mathbf{m}_2$ d'un k-échange précédent qui est non couplé et envoyé à q tel que $\mathbf{m}_2 \overset{\text{SS}}{\dashrightarrow} \mathbf{m}'$ dans $\mathsf{GD}(e_1 \cdots e_n)$. Puisque $\mathbf{m}_1$ est un message non couplé à q et $\mathbf{m}_2$ est non couplé envoyé à q, d'après la RÈGLE 4 de la Figure 4.1, $\mathbf{m}_1 \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_2$. Tout assemblé, on obtient alors $\mathbf{m}_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_1 \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_2 \overset{\text{SS}}{\dashrightarrow} \mathbf{m}'$ ce qui conclut ce cas.

- ✻ $p = \mathsf{proc}_{\mathtt{X}}(\mathbf{m})$, $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \pi$, et $\mathtt{X} = \mathtt{S}$. Alors $\mathbf{m} = \mathbf{m}_{\mathsf{start}}$, ce qui conclut ce cas.

✢ À l'inverse, montrons maintenant que pour tout $\mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}$ et $\mathbf{m}$ tel que $\mathbf{m}_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}$ dans $\mathsf{GD}(e_1 \cdots e_n)$, $\mathsf{proc}_{\mathtt{S}}(\mathbf{m}) \neq \pi$, et $\mathbf{m} \cap \mathtt{X} \neq \emptyset$, on considère que $\mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \in C_X^{\pi\prime}$ (le cas d'angle à prouver $(\mathbf{m}, \mathtt{X}) = (\mathbf{m}_{\mathsf{start}}, \mathtt{S})$, est traité dans le dernier point). Encore une fois, nous raisonnons par récurrence sur le nombre n de k-échanges. Si $n = 0$, il est évident qu'il n'y a pas de tel $\mathbf{m}, \mathtt{X}$. Supposons que cette propriété marche pour tous choix de $\mathbf{m}_1, \mathtt{X}_1$ tels que $\mathbf{m}_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_1$ dans $\mathsf{GD}(e_1 \cdots e_{n-1})$, $\mathsf{proc}_{\mathtt{S}}(\mathbf{m}_1) \neq \pi$, et $\mathbf{m}_1 \cap \mathtt{X}_1 \neq \emptyset$. Soient $\mathbf{m}, \mathtt{X}$ fixés avec $\mathbf{m}_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}$ dans $\mathsf{GD}(e'_1 \cdots e'_n)$, et $\mathbf{m} \cap X \neq \emptyset$, et montrons que $\mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \in C_{\mathtt{X}}^{\pi\prime}$. Nous raisonnons par analyse de cas sur les occurrences dans e_n, ou non, des sommets $\mathbf{m}_{\mathsf{start}}$ et $\mathbf{m}$.

- ✻ $\mathbf{m}_{\mathsf{start}}$ et $\mathbf{m}$ sont dans e_n. Alors comme $\mathbf{m}_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}$ dans $\mathsf{GD}(e_1 \cdots e_n)$ et d'après la preuve du Lemme 4.2.1, nous avons que $\mathbf{m}_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}$ dans $\mathsf{GD}(e_n, \mathcal{B})$. Par définition de $\mathcal{C}_{\mathtt{S}}^{\pi\prime}$ (première ligne), cet ensemble contient $\mathsf{proc}_{\mathtt{X}}(\mathbf{m})$
- ✻ $\mathbf{m}_{\mathsf{start}}$ dans e_n et $\mathbf{m}$ dans $e_1 \cdots e_{n-1}$. Alors il existe $\mathbf{m}_1, \mathbf{m}_2, q$ tels que
  - ✻ $\mathbf{m}_1$ est dans e_n, et soit $\mathbf{m}_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_1$ dans $\mathsf{GD}(e_1 \cdots e_n)$ soit $\mathbf{m}_1 = \mathbf{m}_{\mathsf{start}}$,
  - ✻ $\mathbf{m}_2$ est dans $e_1 \cdots e_{n-1}$, $\mathbf{m}_1 \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_2$ par la RÈGLE 4 de la Figure 4.1, c'est-à-dire, $\mathbf{m}_1$ est un message couplé envoyé à q et $\mathbf{m}_2$ est un message non couplé envoyé à q
  - ✻ soit $\mathbf{m}_2 \overset{\text{XS}}{\dashrightarrow} \mathbf{m}$ dans $\mathsf{GD}(e_1 \cdots e_{n-1}$, ou $\mathbf{m}_2 = \mathbf{m}$

  D'après le premier point, par la preuve du Lemme 4.2.1, on a soit $\mathbf{m}_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_1$ dans $\mathsf{GD}(e_n, \mathcal{B})$ soit $\mathbf{m} = \mathbf{m}_1$. D'après le deuxième point, on a $\mathbf{m}_1 \overset{\text{SS}}{\longrightarrow} \psi_q$ dans $\mathsf{GD}(e_n, \mathcal{B})$. Réunis, on obtient que $\psi_{\mathsf{start}} \overset{\text{SS}}{\dashrightarrow} \psi_p$ dans $\mathsf{GD}(e_n, \mathcal{B}, \mathcal{B}_\pi)$. Par définition de $C_{\mathtt{X}}^{\pi}$, on a donc $C_{\mathtt{X},q} \subseteq C_{\mathtt{X}}^{\pi}$. Et d'après le troisième point, on a $\mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \in C_{\mathtt{X},q}$. Et donc, finalement, $\mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \in C_{\mathtt{X}}^{\pi}$.
- ✻ $\mathbf{m}_{\mathsf{start}}$ dans $e_1 \cdots e_{n-1}$ et $\mathbf{m}$ dans e_n. Alors il y a $\mathbf{m}_1, \mathbf{m}_2, \mathtt{Y}, \mathtt{Z}$ tels que
  - ✻ soit $\mathbf{m}_{\mathsf{start}} \overset{\text{SY}}{\dashrightarrow} \mathbf{m}_1$ dans $\mathsf{GD}(e_1 \cdots e_{n-1})$, soit $(\mathbf{m}, \mathtt{S}) = (\mathbf{m}_1, \mathtt{Y})$
  - ✻ $\mathbf{m}_1 \overset{\text{YZ}}{\longrightarrow} \mathbf{m}_2$
  - ✻ soit $\mathbf{m}_2 \overset{\text{ZS}}{\dashrightarrow} \mathbf{m}$ dans $\mathsf{GD}(e_1 \cdots e_n)$, avec $\mathbf{m}_2$ et $\mathbf{m}$ dans e_n, ou $(\mathbf{m}_2, Z) = (\mathbf{m}, \mathtt{S})$.

  Par le premier point, et par hypothèse de récurrence, nous avons $\mathsf{proc}_{\mathtt{Y}}(m_1) \in C_{\mathtt{X}}^{\pi}$. Par le deuxième point, on a $\mathsf{proc}_{\mathtt{Y}}(\mathbf{m}_1) = \mathsf{proc}_{\mathtt{Z}}(\mathbf{m}_2)$, et d'après la définition des arcs sortant de ψ_{start} on a $\psi_{\mathsf{start}} \overset{\text{SZ}}{\longrightarrow} \mathbf{m}_2$ dans $\mathsf{GD}(e_n, \mathcal{B}, \mathcal{B}_\pi)$. Par le troisième point et la preuve du Lemme 4.2.1, on a soit $\mathbf{m}_2 \overset{\text{ZS}}{\dashrightarrow} \mathbf{m}$ dans $\mathsf{GD}(e_n, \mathcal{B})$ ou $(\mathbf{m}_2, \mathtt{Z}) = (\mathbf{m}, \mathtt{S})$.

Réunis, on a donc $\psi_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}$ dans $\mathsf{GD}(e_n, \mathcal{B}, \mathcal{B}_\pi)$. Par définition de $C_{\mathtt{S}}^{\pi\prime}$ (première ligne), cet ensemble contient $\mathsf{proc}_{\mathtt{X}}(\mathbf{m})$.

- ✻ $\mathbf{m}_{\mathsf{start}}$ et $\mathbf{m}$ dans $e_1 \cdots e_{n-1}$. Si $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}$ dans $\mathsf{GD}(e_1 \cdots e_{n-1})$, alors $\mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \in \mathcal{C}_{\mathtt{R}}^{\pi}$, par hypothèse de récurrence. Sinon, il existe $\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, \mathbf{m}_4, \mathtt{Y}, \mathtt{Z}, q$ tels que
  - ✻ soit $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SY}}{\dashrightarrow} \mathbf{m}_1$ dans $\mathsf{GD}(e_1 \cdots e_{n-1})$, soit $(\mathbf{m}, \mathtt{S}) = (\mathbf{m}_1, \mathtt{Y})$
  - ✻ $\mathbf{m}_1 \overset{\mathrm{YZ}}{\longrightarrow} \mathbf{m}_2$
  - ✻ soit $\mathbf{m}_2 \overset{\mathrm{ZS}}{\dashrightarrow} \mathbf{m}_3$ dans $\mathsf{GD}(e_1 \cdots e_n$, avec $\mathbf{m}_2$ et $\mathbf{m}_3$ dans e_n, soit $(\mathbf{m}_2, Z) = (\mathbf{m}_3, \mathtt{S})$
  - ✻ $\mathbf{m}_3 \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}_4$ à cause de la RÈGLE 4 de la Figure 4.1, c'est-à-dire, $\mathbf{m}_3$ est un message couplé envoyé à q et $\mathbf{m}_4$ est un message non couplé envoyé à q
  - ✻ soit $\mathbf{m}_4 \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}$ dans $\mathsf{GD}(e_1 \cdots e_{n-1})$, soit $(\mathbf{m}_4, \mathtt{S}) = (\mathbf{m}, \mathtt{S})$

  Du premier point, et par hypothèse de récurrence, on déduit $\mathsf{proc}_{\mathtt{Y}}(\mathbf{m}_1) \in C_{\mathtt{X}}^{\pi}$. Du deuxième point, on déduit $\mathsf{proc}_{\mathtt{Y}}(\mathbf{m}_1) = \mathsf{proc}_{\mathtt{Z}}(\mathbf{m}_2)$, et, d'après la définition des arcs sortant de ψ_{start}, on a $\psi_{\mathsf{start}} \overset{\mathrm{SZ}}{\longrightarrow} \mathbf{m}_2$ dans $\mathsf{GD}(e_n, \mathcal{B}, \mathcal{B}_\pi)$. Du troisième point et de la preuve du Lemme 4.2.1, on déduit que soit $\mathbf{m}_2 \overset{\mathrm{ZS}}{\dashrightarrow} \mathbf{m}_3$ dans $\mathsf{GD}(e_n, \mathcal{B})$ soit $(\mathbf{m}_2, Z) = (\mathbf{m}_3, \mathtt{S})$. Du quatrième point, on déduit $\mathbf{m}_3 \overset{\mathrm{SS}}{\dashrightarrow} \psi_q$ dans $\mathsf{GD}(e_n, \mathcal{B})$. Pour résumer, on a $\psi_{\mathsf{start}} \overset{\mathrm{SS}}{\longrightarrow} \psi_q$ dans $\mathsf{GD}(e_n, \mathcal{B}, \mathcal{B}_\pi)$. Par définition de $C_{\mathtt{X}}^{\pi}$, on a donc $C_{\mathtt{X},q} \subseteq C_{\mathtt{X}}^{\pi}$. Du dernier point et de la preuve du Lemme 4.2.1, on déduit que $\mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \in C_{\mathtt{X},q}$, ce qui conclut ce cas.
- ✢ Enfin, terminons cette preuve par l'implication inverse et montrons que $\mathsf{proc}_{\mathtt{S}}(\mathbf{m}_{\mathsf{start}}) \in \mathcal{C}_{\mathtt{S}}^{\pi}$. C'est une conséquence immédiate de la définition de $\mathcal{C}_{\mathtt{S}}^{\pi}$ (voir l'ensemble $\{\mathsf{proc}_{\mathtt{S}}(\mathbf{m}) \mid \mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \pi \ \&\ \mathtt{X} = \mathtt{S}\}$).

Nous avons donc prouvé que

$$C_{\mathtt{X}}^{\pi} = \{\mathsf{proc}_{\mathtt{X}}(\mathbf{m}) \mid (\mathbf{m} \cap \mathtt{X} \neq \emptyset \ \&\ \mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m} \text{ dans } \mathsf{GD}(e')) \text{ ou } (\mathbf{m}, \mathtt{X}) = (\mathbf{m}_{\mathsf{start}}, \mathtt{S})\}$$

Nous pouvons donc conclure cette partie de la preuve.

Soient e' et $e'' = e_1 \cdots e_n \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$ avec $e_1, \cdots e_n \in \mathtt{S}^{\leq k}\mathtt{R}^{\leq k}$ fixés tels que $msc(e') = msc(e'')$.

$\Leftarrow$ Supposons que e' est une exécution k-synchronisable et faisable de $\mathcal{S}'$ et montrons que

$$(\vec{\ell_0}, \mathcal{B}_\emptyset, (\emptyset, \emptyset), \bot) \overset{e_1,k}{\underset{\mathsf{feas}}{\Longrightarrow}} \ldots \overset{e_n,k}{\underset{\mathsf{feas}}{\Longrightarrow}} (\vec{\ell}', \mathcal{B}', \mathcal{B}'_\pi, \mathsf{dest}_\pi)$$

pour certains $\mathcal{B}', \mathcal{B}'_\pi, \mathsf{dest}_\pi$ avec $\mathsf{dest}_\pi \notin \mathcal{C}_{\mathtt{R}}^{\pi}$. Par définition de la relation $\overset{e,k}{\underset{\mathsf{feas}}{\Longrightarrow}}$, $\mathcal{B}'$, $\mathcal{B}_\pi$ et dest_π sont déterminés de façon unique et il est alors suffisant de prouver que $\mathsf{dest}_\pi \notin \mathcal{C}_{\mathtt{R}}^{\pi}$. Supposons par l'absurde que $\mathsf{dest}_\pi \in \mathcal{C}_{\mathtt{R}}^{\pi}$. Alors, par la propriété que nous venons de prouver, il existe $\mathbf{m}$ tel que $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \mathsf{dest}_\pi$, $\mathbf{m} \cap \mathtt{R} \neq \emptyset$, et $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}$ dans $\mathsf{GD}(e'_1 \cdots e'_n)$. Donc on a $\mathbf{m}_{\mathsf{start}} \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m} \overset{\mathrm{RR}}{\longrightarrow} \mathbf{m}_{\mathsf{stop}}$ dans $\mathsf{GD}(e')$, et par le Théorème 4.1.1, e' ne devrait pas être faisable ce qui est une contradiction. Finalement, $\pi \notin \mathcal{C}_{\mathtt{R}}^{\pi}$ car e' est une exécution de $\mathcal{S}'$ et donc $msc(e')$ est mb-réalisable.

$\Rightarrow$ Supposons que

$$(\vec{\ell_0}, \mathcal{B}_\emptyset, (\emptyset,\emptyset), \bot) \xrightarrow[\text{feas}]{e_1,k} \dots \xrightarrow[\text{feas}]{e_n,k} (\vec{\ell}', \mathcal{B}', \mathcal{B}_\pi, \mathsf{dest}_\pi).$$

pour certains $\mathcal{B}', \mathcal{B}'_\pi, \mathsf{dest}_\pi$ avec $\pi \notin \mathcal{C}^\pi_{\mathtt{R}}$, et montrons que e' est une exécution k-synchronisable et faisable de $\mathcal{S}'$. Par la définition de $\xrightarrow[\text{feas}]{e,k}$, on a

$$(\vec{\ell_0}, \mathcal{B}_\emptyset) \xrightarrow[\text{real}]{e_1,k} \dots \xrightarrow[\text{real}]{e_n,k} (\vec{\ell}', \mathcal{B}')$$

et d'après le Lemme 4.2.1, $msc(e')$ est k-synchrone.

Puisque les deux dernières actions $\mathtt{s}(\pi,q,m) \cdot \mathtt{r}(\pi,q,m)$ peuvent être placées dans un nouveau k-échange, et n'empêchent pas d'être mb-réalisable (car $\pi \notin \mathcal{C}^\pi_{\mathtt{R}}$), e' est une exécution k-synchronisable du système $\mathcal{S}'$. Il reste alors à prouver que e' est faisable. Raisonnons par contradiction et supposons que e' n'est pas faisable. Par le Théorème 4.1.1, il existe $\mathbf{m}$ tel que $\mathbf{m}_{\mathsf{start}} \xrightarrow{\text{SS}} \mathbf{m} \xrightarrow{\text{RR}} \mathbf{m}_{\mathsf{stop}}$ dans $\mathsf{GD}(e')$. En d'autres mots, $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \mathsf{dest}_\pi$, $\mathbf{m} \cap \mathtt{R} \neq \emptyset$, et $\mathbf{m}_{\mathsf{start}} \xrightarrow{\text{SS}} \mathbf{m}$ dans $\mathsf{GD}(e')$. Donc, par la propriété prouvée ci-dessus, $\mathsf{dest}_\pi \in \mathcal{C}^\pi_{\mathtt{R}}$, ce qui nous amène à une contradiction.

□

Comparaison avec [Bouajjani et al., 2018a] 3. *La définition de livraison causale définie dans [Bouajjani et al., 2018a, Section 3] n'est pas suffisante pour la vérification de la communication en boîte aux lettres et en FIFO. Ceci entraîne des problèmes dans l'ensemble de l'algorithme. Pour vérifier qu'une exécution est faisable, les auteurs utilisent un* moniteur *[Bouajjani et al., 2018a, Section 6.2] qui regarde chaque action de l'exécution et ajoute les processus qui ne sont plus autorisés à envoyer de messages au destinataire de π, correspondant donc à leur définition de livraison causale. On peut citer la construction de cet ensemble $B(p)'$ pour une séquence e (adaptée à notre notation) que l'on peut trouver dans [Bouajjani et al., 2018a, Figure 7] :*

$$\begin{aligned} B(p)' = B(p) \cup \{q \mid \exists \mathtt{s}(p',q',m) \in e \cap \mathtt{S} \wedge \\ (\nexists r, \mathtt{s}(p',q',m) \vdash r \wedge \mathsf{proc}_{\mathtt{S}}(m) = q \wedge \mathsf{proc}_{\mathtt{R}}(m) = p) \vee \\ (\mathsf{proc}_{\mathtt{S}}(m) \in B(p) \wedge \mathsf{proc}_{\mathtt{R}}(m) = q)\} \end{aligned}$$

Cependant, d'après la description de leur moniteur, l'exécution suivante $e' = \mathsf{deviate}(e \cdot r)$, dont le MSC est en Figure 4.14.a, est faisable, ce qui signifie qu'elle peut être jouée dans $\mathcal{S}'$ et que l'exécution $e \cdot r$ peut être jouée dans $\mathcal{S}$.

$$\begin{aligned} e' = &\mathtt{s}(r,\pi,(s,m_1)) \cdot \mathtt{r}(r,\pi,(s,m_1)) \cdot \mathtt{s}(r,t,m_2) \cdot \mathtt{r}(r,t,m_2) \cdot \mathtt{s}(q,t,m_3) \cdot \mathtt{r}(q,t,m_3) \cdot \\ &\mathtt{s}(p,q,m_4) \cdot \mathtt{r}(p,q,m_4) \cdot \mathtt{s}(q,s,m_5) \cdot \mathtt{r}(q,s,m_5) \cdot \mathtt{s}(\pi,s,m_1) \cdot \mathtt{r}(\pi,s,m_1) \end{aligned}$$

En effet, le processus q ne se retrouve pas interdit d'envoi pour le processus s : seuls les processus r et t appartiennent à $B(s)$. Cependant, le graphe de dépendances étendu présenté en Figure 4.14.b montre que cette exécution n'est pas faisable, car il existe une dépendance causale entre m_1 et m_5. On peut également suivre les ensembles $\mathcal{C}^\pi_{\mathtt{S}}$ en orange et $\mathcal{C}^\pi_{\mathtt{R}}$ en vert, on constate également que $s \in \mathcal{C}^\pi_{\mathtt{R}}$.

Dans l'algorithme de décision de [Bouajjani et al., 2018a], cette exécution est considérée comme faisable et donc appartient à $\text{MSC}_k(\mathcal{S})$*. Toutefois, dû au message* m_4 *dont la réception précède l'envoi de* m_5*, elle serait également mauvaise (on constate l'arc RS dans le graphe de dépendances étendu). Ainsi, l'algorithme déterminerait que le système n'est pas* k*-synchronisable dû à l'existence d'une exécution faisable et mauvaise, alors qu'il ne fallait pas la prendre en compte car elle n'existait que dans* $\mathcal{S}'$*. La décision de la* k*-synchronisabilité serait trompée et l'on aurait ici affaire à un faux négatif.*

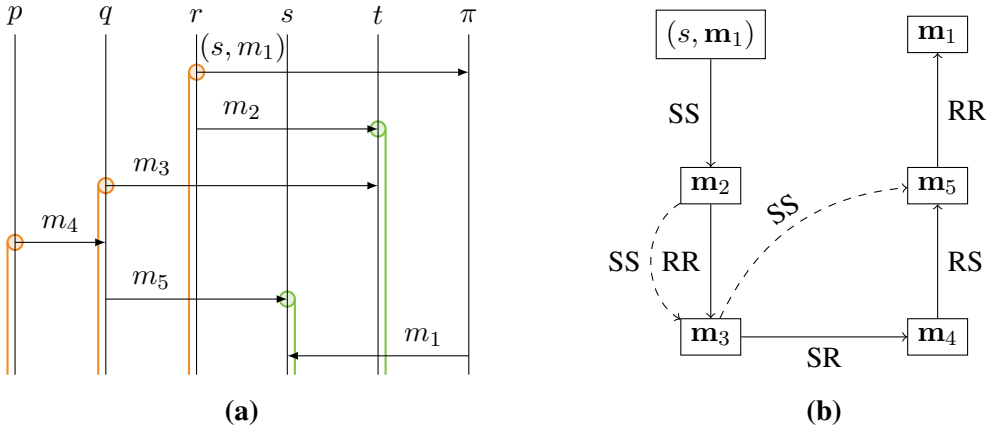


FIGURE 4.14 – Un MSC d'une exécution pas faisable avec en orange les processus de $\mathcal{C}_{\text{S}}^{\pi}$ et en vert ceux de $\mathcal{C}_{\text{R}}^{\pi}$ (a) et le graphe de dépendances étendu associés (b)

4.3.3 Reconnaissance des mauvaises exécutions

La dernière étape consiste donc à construire un automate non déterministe capable de reconnaître les MSC des mauvaises exécutions, autrement dit, les exécutions $e' = \mathsf{deviate}(e \cdot r)$ faisables mais où $e \cdot r$ n'est pas k-synchronisable dans $\mathcal{S}$. Pour cela, nous utiliserons le graphe de dépendances (non étendu), sans les arcs de la forme $\overset{\text{XY}}{\dashrightarrow}$. L'objectif est de reconstituer la composante fortement connexe à laquelle appartient le message dévié dans le graphe de dépendances original et y vérifier sa taille et l'absence d'arc RS.

Soit $\mathsf{Post}^*(\mathbf{m}) = \{\mathbf{m}' \in V \mid \mathbf{m} \to^* \mathbf{m}'\}$ l'ensemble des sommets accessibles depuis $\mathbf{m}$ et $\mathsf{Pre}^*(\mathbf{m}) = \{\mathbf{m}' \in V \mid \mathbf{m}' \to^* \mathbf{m}\}$ l'ensemble des sommets co-accessibles depuis $\mathbf{m}$. Pour un ensemble de sommets $U \subseteq V$, soit $\mathsf{Post}^*(U) = \bigcup\{\mathsf{Post}^*(\mathbf{m}) \mid \mathbf{m} \in U\}$ et $\mathsf{Pre}^*(U) = \bigcup\{\mathsf{Pre}^*(\mathbf{m}) \mid \mathbf{m} \in U\}$.

Pour reconstituer la composante fortement connexe contenant le message dévié, nous regardons alors l'ensemble des messages accessibles depuis $\mathbf{m}_{\mathsf{start}}$ et l'ensemble des messages co-accessibles depuis $\mathbf{m}_{\mathsf{stop}}$. L'union de ces deux ensembles constitue l'ensemble des messages de la composante fortement connexe avec $\mathbf{m}_{\mathsf{start}}$ et $\mathbf{m}_{\mathsf{stop}}$. Ainsi, une exécution sera mauvaise soit si l'ensemble des messages ne peuvent pas être organisés en k-échanges, et donc s'il existe un arc RS dans une composante fortement connexe, soit si un k-échange est trop grand, et donc de taille supérieure à $k + 1$.

Lemme 4.3.4. *Une exécution faisable e' est mauvaise si et seulement si au moins l'une des deux affirmations suivantes est vraie :*

1. $\mathbf{m}_{\mathsf{start}} \rightarrow^* \xrightarrow{\text{RS}} \rightarrow^* \mathbf{m}_{\mathsf{stop}}$
2. *l'ensemble* $\mathsf{Post}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{Pre}^*(\mathbf{m}_{\mathsf{stop}})$ *est de taille supérieure ou égale à* $k+2$.

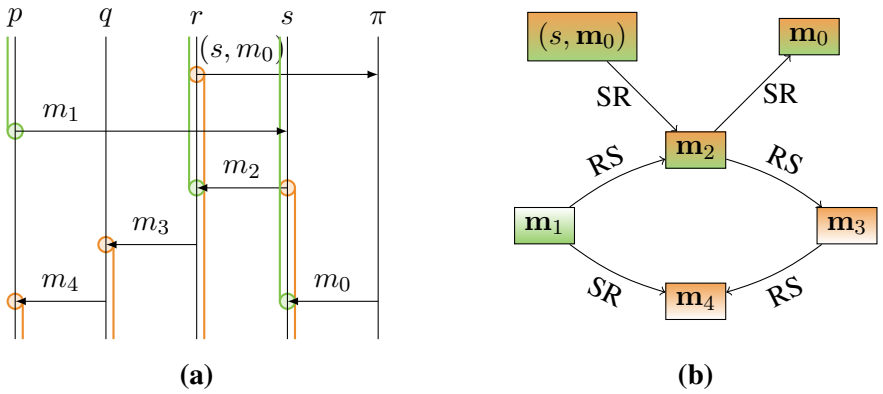
Démonstration.

Puisque $msc(e')$ est k-synchrone et $e' = \mathsf{deviate}(e \cdot r)$, $msc(e)$ (sans la dernière réception r) est k-synchrone. D'après le Théorème 4.1.2, e' est une exécution mauvaise si et seulement si $\mathsf{GD}(e \cdot r)$ contient soit un cycle contenant un arc RS soit une composante fortement connexe de taille $\geq k+1$. Cette composante fortement connexe doit contenir un sommet associé à la dernière réception r de $e \cdot r$. Dans $\mathsf{GD}(e')$, cette composante fortement connexe correspond à l'ensemble des sommets qui sont à la fois accessibles depuis ψ_{start} et co-accessibles depuis ψ_{stop}. Puisque les sommets ψ_{start} et ψ_{stop} comptent pour le même sommet dans le graphe de $e \cdot r$, la taille de la composante fortement connexe correspond à la taille de l'ensemble $\mathsf{Post}^*(\psi_{\mathsf{start}}) \cap \mathsf{Pre}^*(\psi_{\mathsf{stop}})$ moins un.

□

Afin de déterminer si un message donné $\mathbf{m}$ de $\mathsf{GD}(e')$ doit être considéré comme accessible (ou co-accessible), on calculera à l'entrée et à la sortie de chaque k-échange quels processus sont accessibles ou co-accessibles.

Exemple 4.3.4 – Considérons le MSC de la Figure 4.15.a composé de 5 1-échanges. Nous représentons en orange le fait d'être accessible. Au moment de l'envoi de (s, m_0), qui correspond au sommet $\mathbf{m}_{\mathsf{start}}$, le processus r devient accessible : tous les sommets des messages qui impliqueront r seront accessibles depuis le sommet $\mathbf{m}_{\mathsf{start}}$ dans le graphe de dépendances. À l'envoi de m_2, le processus s devient à son tour accessible, car le processus r sera accessible au moment de la réception de m_2. De la même façon, q devient accessible après avoir reçu m_3 car r était accessible au moment de l'envoi de m_3, et p devient accessible après avoir reçu m_4 car q était accessible quand il l'a envoyé.



(a) **(b)**

FIGURE 4.15 – Un MSC avec déviation (a) et son graphe de dépendances (b) avec spécifications des processus accessibles en orange et co-accessibles en vert

La co-accessibilité fonctionne de la même façon, mais notre raisonnement doit remonter le temps. Nous la représentons en vert. Le processus s arrête d'être co-accessible quand il reçoit m_0, le processus r arrête d'être co-accessible après avoir reçu m_2 et le processus p arrête d'être co-accessible en envoyant m_1.

Le graphe de dépendances représenté sur la Figure 4.15.b colore en orange les messages accessibles et en vert les messages co-accessibles. Le seul message qui est envoyé par un processus étant à la fois accessible et co-accessible au moment de l'envoi est le message m_2, il s'agit donc du seul message qui sera compté comme appartenant à la composante fortement connexe.

Plus formellement, posons e une séquence d'actions, $\mathsf{GD}(e)$ son graphe de dépendances et P, Q deux ensembles de processus tels que

$$\begin{aligned}\mathsf{Post}_e(P) =& \mathsf{Post}^*(\{\mathbf{m} \mid \mathsf{procs}(\mathbf{m}) \cap P \neq \emptyset\})\\ \mathsf{Pre}_e(Q) =& \mathsf{Pre}^*(\{\mathbf{m} \mid \mathsf{procs}(\mathbf{m}) \cap Q \neq \emptyset\})\end{aligned}$$

Ces ensembles sont introduits pour représenter la vue locale d'un k-échange des ensembles $\mathsf{Post}^*(\mathbf{m}_{\mathsf{start}})$ et $\mathsf{Pre}^*(\mathbf{m}_{\mathsf{stop}})$. Par exemple, pour une exécution e correspondant au MSC de la Figure 4.15, on aurait

$$\begin{aligned}\mathsf{Post}_e(\{\pi\}) =& \{(s, \mathbf{m}_0), \mathbf{m}_2, \mathbf{m}_3, \mathbf{m}_4, \mathbf{m}_0\} \text{ et}\\ \mathsf{Pre}_e(\{\pi\}) =& \{\mathbf{m}_0, \mathbf{m}_2, \mathbf{m}_1, (s, \mathbf{m}_0)\}\end{aligned}$$

Pour chaque k-échange e_i, la taille de l'intersection entre $\mathsf{Post}_{e_i}(P)$ et $\mathsf{Pre}_{e_i}(Q)$ donne le nombre de messages qui participent à la composante fortement connexe dans le k-échange courant, ce qui permettra de calculer sa taille globale. Dans la transition $\xrightarrow[\mathsf{bad}]{e,k}$, cette valeur est stockée dans la variable `cnt`.

La dernière chose à considérer est l'apparition d'un arc RS au sein de cette composante fortement connexe. Pour cela, nous utilisons la fonction $\texttt{lastisRec} : \mathbb{P} \to \{\mathsf{True}, \mathsf{False}\}$, qui, pour chaque processus, garde en mémoire si la dernière action effectuée dans le k-échange précédent était une réception. Pour rappel, il ne peut pas y avoir d'arc RS au sein d'un k-échange par définition même d'un k-échange. Les seuls qui pourraient apparaître au sein de la composante fortement connexe serait alors à la frontière entre deux k-échanges. Cependant, les messages appartenant à la composante fortement connexe ne peuvent être connus qu'à la fin de l'exécution. Il faut donc alors se souvenir, pour chaque processus, si l'on a croisé un arc RS pour analyser cette information plus tard : on considère donc une variable `sawRS` qui indique la présence d'un tel arc sur un processus. Pour chaque processus, elle est mise à jour en fonction de la valeur de `lastisRec` et de la première action du processus concerné dans le k-échange courant.

La relation de transition $\xrightarrow[\mathsf{bad}]{e,k}$ définie dans la Figure 4.16 manipule des configurations abstraites de la forme suivante :

$$(P, Q, \texttt{cnt}, \texttt{sawRS}, \texttt{lastisRec})$$

où $P, Q \subseteq \mathbb{P}$, `sawRS` est une valeur booléenne et `cnt` est un compteur borné à $k + 2$. On note $\texttt{lastisRec}_0$ la fonction où $\texttt{lastisRec}(p) = \mathsf{False}$ pour tout $p \in \mathbb{P}$.

$$P' = \mathsf{procs}(\mathsf{Post}_e(P)) \qquad Q = \mathsf{procs}(\mathsf{Pre}_e(Q')) \qquad SCC_e = \mathsf{Post}_e(P) \cap \mathsf{Pre}_e(Q')$$
$$\mathtt{cnt}' = \min(k+2, \mathtt{cnt}+n) \quad \text{où } n = |SCC_e|$$
$$\mathtt{lastisRec}'(q) \Leftrightarrow (\exists \mathbf{m}.\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = q \wedge \mathbf{m} \cap \mathtt{R} \neq \emptyset) \vee$$
$$(\mathtt{lastisRec}(q) \wedge \nexists m \in \mathbb{V}.\mathsf{proc}_{\mathtt{S}}(\mathbf{m}) = q)$$
$$\mathtt{sawRS}' = \mathtt{sawRS} \vee (\exists \mathbf{m})(\exists p \in \mathbb{P} \setminus \{\pi\})\ \mathsf{proc}_{\mathtt{S}}(\mathbf{m}) = p \wedge \mathtt{lastisRec}(p) \wedge p \in P \cap Q$$

$$(P, Q, \mathtt{cnt}, \mathtt{sawRS}, \mathtt{lastisRec}) \xRightarrow[\mathsf{bad}]{e,k} (P', Q', \mathtt{cnt}', \mathtt{sawRS}', \mathtt{lastisRec}')$$

FIGURE 4.16 – Définition de la relation de transition $\xRightarrow[\mathsf{bad}]{e,k}$

Lemme 4.3.5. *Soit e' une exécution faisable et k-synchronisable de $\mathcal{S}'$. Alors, e' est mauvaise si et seulement s'il existe $e'' = e_1 \cdots e_n \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$ telle que*

$$(\{\pi\}, Q, 0, \mathsf{False}, \mathtt{lastisRec}_0) \xRightarrow[\mathsf{bad}]{e_1,k} \cdots \xRightarrow[\mathsf{bad}]{e_n,k} (P', \{\pi\}, \mathtt{cnt}, \mathtt{sawRS}, \mathtt{lastisRec})$$

avec une des deux affirmations suivantes vraie :

$\mathtt{sawRS} = \mathsf{True}$ *ou* $\mathtt{cnt} = k+2$

et $e_1, \cdots, e_n \in \mathtt{S}^{\leq k}\mathtt{R}^{\leq k}$ *et* $msc(e') = msc(e''), P', Q \subseteq \mathbb{P}, \mathtt{sawRS} \in \{\mathsf{True}, \mathsf{False}\}, \mathtt{cnt} \in \{0, \cdots, k+2\}$.

Démonstration.

Soit e' une exécution k-synchronisable faisable et mauvaise telle que $e'' = e_1 \cdots e_n \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$ avec $msc(e') = msc(e'')$. Nous montrons que

$$(\{\pi\}, Q, \mathsf{False}, 0) \xRightarrow[\mathsf{bad}]{e_1,k} \cdots \xRightarrow[\mathsf{bad}]{e_n,k} (P', \{\pi\}, \mathtt{sawRS}, \mathtt{cnt})$$

pour certains P', Q et avec soit $\mathtt{sawRS} = \mathsf{True}$ soit $\mathtt{cnt} = k+2$. Nous procédons par récurrence sur n.

Base $n = 2$ Notons que, pour une exécution faisable, il y a au moins deux k-échanges car la déviation ne peut se traduire en un seul k-échange : l'envoi du processus π au destinataire d'origine doit être après l'envoi (et la réception) de l'expéditeur d'origine au processus π et doit donc appartenir à un k-échange différent et ultérieur.
Alors, $e'' = e_1 \cdot e_2$ et on montre alors que

$$(\{\pi\}, Q, \mathsf{False}, 0) \xRightarrow[\mathsf{bad}]{e_1,k} (P', Q', \mathtt{sawRS}', \mathtt{cnt}) \xRightarrow[\mathsf{bad}]{e_2,k} (P'', \{\pi\}, \mathtt{sawRS}, \mathtt{cnt}')$$

Par le Lemme 4.3.4, on sait que soit $\psi_{\mathsf{start}} \rightarrow^* \xrightarrow{\text{RS}} \rightarrow^* \psi_{\mathsf{stop}}$, soit la taille de l'ensemble $\mathsf{Post}^*(\psi_{\mathsf{start}}) \cap \mathsf{Pre}^*(\psi_{\mathsf{stop}})$ est plus grand ou égal à $k+2$.

- ✢ Si $\psi_{\mathsf{start}} \rightarrow^* \xrightarrow{\text{RS}} \rightarrow^* \psi_{\mathsf{stop}}$, alors puisqu'un label RS ne peut exister dans un graphe de dépendances local, il existe deux chemins $\psi_{\mathsf{start}} \rightarrow^* \mathbf{m}_1$ dans $\mathsf{GD}(e_1)$ et $\mathbf{m}_2 \rightarrow^* \psi_{\mathsf{stop}}$ dans $\mathsf{GD}(e_2)$, avec $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}_1) = \mathsf{proc}_{\mathtt{S}}(\mathbf{m}_2)$. On a alors que $\mathbf{m}_2 \in \mathsf{Pre}_{e_2}(\pi)$ et $\mathtt{lastisRec}(\mathsf{proc}_{\mathtt{S}}(\mathbf{m}_2))$ est True, et donc $\mathtt{sawRS}$ devient True, concluant cette partie de la preuve.

- ✢ Maintenant, supposons que la taille de l'ensemble $\mathsf{Post}^*(\psi_{\mathsf{start}}) \cap \mathsf{Pre}^*(\psi_{\mathsf{stop}})$ est supérieure ou égale à $k+2$. Nous montrons alors que tous les sommets de $\mathsf{Post}^*(\psi_{\mathsf{start}}) \cap \mathsf{Pre}^*(\psi_{\mathsf{stop}})$ ont été comptés soit dans le premier k-échange soit dans le second. Prenons $\mathbf{m} \in \mathsf{Post}^*(\psi_{\mathsf{start}}) \cap \mathsf{Pre}^*(\psi_{\mathsf{stop}})$ alors il existe un chemin $\psi_{\mathsf{start}} \to^* v \to^* \psi_{\mathsf{stop}}$ et $\mathbf{m}$ est un message qui appartient soit au premier soit au second k-échange.
  - ✻ Si $\mathbf{m}$ appartient au premier, alors on peut diviser le chemin en deux parties telles que $\psi_{\mathsf{start}} \to^* v \to^* \mathbf{m}_1$ est dans $\mathsf{GD}(e_1)$, $\mathbf{m}_2 \to^* \psi_{\mathsf{stop}}$ est dans $\mathsf{GD}(e_2)$ et $\mathsf{procs}(\mathbf{m}_1) \cup \mathsf{procs}(\mathbf{m}_2) = \{p\} \neq \emptyset$. À partir de là, on peut déduire que le processus $p \in Q'$ et donc $\mathbf{m} \in \mathsf{Pre}_{e_1}(Q')$. De plus, $\mathbf{m} \in \mathsf{Post}_{e_1}(\pi)$ et donc le sommet $\mathbf{m}$ est compté dans le premier k-échange.
  - ✻ De la même façon, si $\mathbf{m}$ appartient au second k-échange, on peut diviser le chemin en deux parties telles que $\psi_{\mathsf{start}} \to^* \mathbf{m}_1$ est dans $\mathsf{GD}(e_1)$, $\mathbf{m}_2 \to^* \mathbf{m} \to^* \psi_{\mathsf{stop}}$ est dans $\mathsf{GD}(e_2)$ et $\mathsf{procs}(\mathbf{m}_1) \cup \mathsf{procs}(\mathbf{m}_2) = \{p\} \neq \emptyset$. À partir de là, on peut déduire que $p \in P'$ et donc $\mathbf{m} \in \mathsf{Post}_{e_2}(P')$. De plus, $\mathbf{m} \in \mathsf{Pre}_{e_2}(\pi)$ et donc le sommet $\mathbf{m}$ est compté dans le deuxième k-échange.

  Ainsi, tous les sommets de $\mathsf{Post}^*(\psi_{\mathsf{start}}) \cap \mathsf{Pre}^*(\psi_{\mathsf{stop}})$ sont considérés et si $\mathsf{Post}^*(\psi_{\mathsf{start}}) \cap \mathsf{Pre}^*(\psi_{\mathsf{stop}}) \geq k+2$ et donc la variable $\mathtt{cnt}' \geq k+2$, ce qui conclut cette partie de la preuve.

Récurrence C'est une généralisation facile de ce qui a été dit dans la partie précédente de la preuve. En considérant que, par hypothèse de récurrence, les ensembles $\mathsf{Post}_{e_i}(P)$ et $\mathsf{Pre}_{e_i}(Q)$ contiennent respectivement tous les processus qui sont accessibles depuis le message envoyé au processus π et co-accessibles depuis le message envoyé au destinataire initial par le processus π.

$\Leftarrow$ Soit e' une exécution k-synchronisable et faisable, $e'' = e_1 \cdots e_n \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$ avec $msc(e') = msc(e'')$, et $P', Q \subseteq \mathbb{P}$, $\mathtt{cnt} \in \{0, \ldots, k+2\}$ tels que

$$(\{\pi\}, Q, 0) \xrightarrow[\mathsf{bad}]{e_1,k} \cdots \xrightarrow[\mathsf{bad}]{e_n,k} (P', \{\pi\}, \mathtt{cnt})$$

On suppose que $\mathtt{cnt} = k+2$. e' est faisable par le Lemme 4.3.3. Chaque sommet $\mathbf{m}$ appartient à $\mathsf{Pre}_{e_i}(P_i) \cap \mathsf{Post}_{e_i}(Q'_i) \setminus \mathbf{m}_{\mathsf{start}}$ mais également à $\mathsf{Post}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{Pre}^*(\mathbf{m}_{\mathsf{stop}})$ alors $|\ \mathsf{Post}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{Pre}^*(\mathbf{m}_{\mathsf{stop}})\ | \geq k+2$. Alors, e' est mauvaise, ce qui est conclut cette preuve pour la communication en boîte aux lettres.

□

Comparaison avec [Bouajjani et al., 2018a] 4. *Comme pour la notion de faisabilité, pour déterminer si une exécution est mauvaise, les auteurs utilisent un moniteur [Bouajjani et al., 2018a, Figure 10] qui construit un chemin entre l'envoi au processus π et l'envoi final au destinataire original. Aux problèmes déjà observés précédemment s'ajoute le fait que ce moniteur peut non seulement détecter un arc RS alors qu'il n'y en a pas, mais aussi manquer des arcs RS existants. Le fonctionnement est le suivant : les messages sont observés, dans un ordre non déterministe, en mettant à jour des variables. La variable* conflict *se souvient du dernier processus engagé, elle est initialisée avec l'expéditeur du message envoyé à π. C'est sa gestion qui crée les erreurs, elle se*

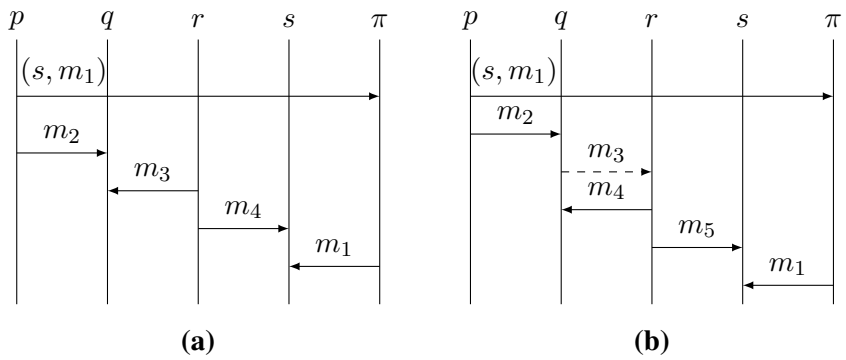


FIGURE 4.17 – Des MSC entraînant des faux positifs (a) ou des faux négatifs (b) avec l'algorithme de [Bouajjani et al., 2018a]

souvient du dernier processus engagé, qui est donc l'un des processus du message choisi de façon non déterministe. Une variable lastisrec *fonctionne de façon similaire à la nôtre : elle est vraie si la dernière action observée était une réception, et est comparée avec l'action du message en cours pour savoir si un arc RS est présent, indiqué alors par la variable* sawRS. *Les exemples suivants mettent en lumière les cas où ce moniteur donne de mauvaises réponses. Une variable compte également le nombre de message vus, pour construire le chemin entre* $\mathbf{m}_{\mathsf{start}}$ *et* $\mathbf{m}_{\mathsf{stop}}$*, mais, étant donné qu'elle recherche un cycle plutôt qu'une composante fortement connexe, nous n'avons pas besoin de nous y attarder.*

La Figure 4.17.a montre un MSC associé à une exécution faisable qui ne contient pas d'arc RS. À la lecture du message m_2*, les variables sont les suivantes :* conflict $= q$ *(autrement* m_3 *n'a pas de processus en commun et on ne peut pas continuer le chemin),* lastisrec $=$ True. *Le moniteur tel que défini dans [Bouajjani et al., 2018a, Figure 10] considère*

- ✢ *soit la réception de* m_3 *auquel cas* conflict $= q$, lastisrec $=$ True *et donc* sawRS $=$ False,
- ✢ *soit l'envoi de* m_3 *et alors* conflict $= r$, lastisrec $=$ False *mais alors* sawRS $=$ True.

Le deuxième cas seulement permettra de lire le message m_4 *donc le premier est ignoré. Un arc étiqueté RS est alors, à tord, détecté. À l'inverse, la Figure 4.17.b montre un MSC associé à une exécution faisable mais mauvaise. Cependant, avec le moniteur de [Bouajjani et al., 2018a, Figure 10], après l'envoi du message* m_3*, on a* conflict $= q$, lastisrec $=$ False *et donc* sawRS $=$ False. *L'on peut regarder le message* m_4*. Il faut avoir* conflict $= r$*, sinon, le chemin ne pourra être continué avec* m_5*. Donc, l'on obtient* conflict $= r$, lastisrec $=$ False *et donc* sawRS $=$ False. *Ainsi, la réception de* m_4 *est ignoré et l'arc RS existant est ignoré au profit d'un arc SS qui est en réalité absent.*

Enfin, nous pouvons alors conclure par le théorème suivant.

Théorème 4.3.6. *La k-synchronisabilité pour un système $\mathcal{S}$ communiquant en boîte aux lettres est décidable pour un k donné.*

Démonstration.

Soit $\mathcal{S}$ un système donné. Par les Lemmes 4.3.1, 4.4.4 et 4.3.5, $\mathcal{S}$ n'est pas k-synchronisable si et seulement s'il existe une séquence d'action $e' = e'_1 \cdots e'_n \cdot s \cdot r$ tel que $e_i \in \mathtt{S}^{\leq k}\mathtt{R}^{\leq k}$, $s = \mathtt{s}(\pi, q, m)$, $r = \mathtt{r}(\pi, q, m)$,

$$(\vec{\ell_0}, \mathcal{B}_\emptyset, (\emptyset, \emptyset), \bot) \xRightarrow[\mathsf{feas}]{e'_1,k} \cdots \xRightarrow[\mathsf{feas}]{e'_n,k} (\vec{\ell}\,', \mathcal{B}', \mathcal{B}'_\pi, q)$$

et

$$(\{\pi\}, Q, \mathsf{False}, 0) \xRightarrow[\mathsf{bad}]{e'_1,k} \cdots \xRightarrow[\mathsf{bad}]{e'_n,k} \xRightarrow[\mathsf{bad}]{s\cdot r,k} (P', \{\pi\}, \mathtt{sawRS}, \mathtt{cnt})$$

pour certain $\vec{\ell}\,', \mathcal{B}', \mathcal{B}'_\pi, Q, P'$ avec $\mathsf{dest}_\pi \notin \mathcal{C}^\pi_\mathtt{R}$. Puisque $\xRightarrow[\mathsf{feas}]{e,k}$ et $\xRightarrow[\mathsf{bad}]{e,k}$ sont des relations sur des ensembles finis, l'existence d'une telle séquence d'actions est décidable.

□

4.4 Adaptation aux systèmes communiquant en pair à pair

Dans cette section, nous nous intéressons donc à la k-synchronisabilité appliquée à un système communiquant en pair à pair. Aucune étude de systèmes communiquant en pair à pair n'est faite dans [Bouajjani et al., 2018a]. Nous relèverons alors seulement les différences que l'on peut constater avec les sytèmes communicants en boîte aux lettres que nous venons d'étudier. Ceux-ci résident principalement dans la gestion de la réalisabilité qui impose bien moins de contraintes dans les systèmes en pair à pair qu'en boîte aux lettres. La détection d'un MSC pp-réalisable en est donc simplifiée.

La preuve de la k-synchronisabilité pour un k donné est similaire à celle effectuée pour les systèmes en boîte aux lettres, il s'agit simplement d'une adaptation, et les étapes sont donc les mêmes. Cette section se compose ainsi, dans un premier temps, de la preuve de décidabilité de l'accessibilité, nécessaire pour le deuxième temps, étant la preuve de la k-synchronisabilité pour un k donné.

4.4.1 Caractérisation graphique

Commençons par la reconnaissance d'un MSC pp-réalisable. Comme pour les systèmes en boîte aux lettres, si un message est non couplé dans le k-échange courant, il ne sera jamais reçu. De plus, après qu'un processus p ait envoyé un message non couplé à un processus q, p n'est plus autorisé à envoyer un message couplé à q. Cependant, être pp-réalisable est plus simple que d'être mb-réalisable et pour conséquence, le graphe de dépendances étendu n'est plus utile. Les sommets de synthèse et les arcs étendus ne sont plus nécessaires : toutes les informations nécessaires peuvent être contenus dans une fonction $\mathcal{B}^\mathsf{p}(p)$ retenant quels processus ne peuvent plus envoyer de messages couplés au processus p. Nous la verrons en détail dans la prochaine section.

Poursuivons avec la caractérisation d'une exécution k-synchronisable qui est la même que pour les systèmes en boîte aux lettres. En effet, le mode de communication n'a ici pas d'impact. La preuve est alors identique à celle du Théorème 4.1.2.

Théorème 4.4.1. *Soit μ un MSC* pp*-réalisable. μ est k-synchrone si et seulement si toute composante fortement connexe du graphe de dépendances est de taille inférieure ou égale à k et ne contient pas d'arc RS.*

4.4.2 La décidabilité de l'accessibilité

Pour établir la décidabilité de l'accessibilité pour les systèmes en pair à pair k-synchronisables, on définit une relation de transition $\xrightarrow[\mathsf{real\cdot pp}]{e,k}$, en Figure 4.18, pour une séquence d'actions e décrivant un k-échange. Elle s'appuie sur la fonction $\mathcal{B}^{\mathsf{p}}$ introduite de façon intuitive précédemment et définie formellement dans cette relation de transition. Elle est nécessaire pour vérifier que la séquence de k-échanges est pp-réalisable. Plus précisément, à chaque action d'envoi, si le message n'est pas couplé, l'expéditeur est ajouté à la fonction $\mathcal{B}^{\mathsf{p}}$ du destinataire. On vérifie finalement qu'aucun message couplé n'a son expéditeur dans l'ensemble des processus interdits pour le destinataire. Cette relation est définie avec $(\vec{\ell_0}, \mathcal{B}^{\mathsf{p}}_{\emptyset})$ comme état initial, où $\mathcal{B}^{\mathsf{p}}_{\emptyset} = (\emptyset)_{p\in\mathbb{P}}$. On note $\mathsf{proc}(a)$ le processus faisant l'action a.

$$\frac{\begin{array}{c} e = s_1 \cdots s_m \cdot r_1 \cdots r_{m'} \in \mathtt{S}^m \cdot \mathtt{R}^{m'} \qquad 0 \le m' \le m \le k \\ (\vec{\ell}, \vec{c_\emptyset}) \overset{e}{\Mapsto} (\vec{\ell'}, \vec{c}) \text{ pour certains } \vec{\ell'} \text{ et } \vec{c} \\ (\forall q \in \mathbb{P}) \quad \mathcal{B}^{\mathsf{p}}_{i+1}(q) = \mathcal{B}^{\mathsf{p}}_{i}(q) \cup \{p \mid s_i = \mathtt{s}(p,q,m) \wedge s_i \text{ est non couplé}\} \\ (\forall s_i \in e \cap \mathtt{S}) \quad s_i \mapsto r_j \implies \mathsf{proc}(s_i) \notin \mathcal{B}^{\mathsf{p}}_{i}(\mathsf{proc}(r_j)) \end{array}}{(\vec{\ell}, \mathcal{B}^{\mathsf{p}}) \xrightarrow[\mathsf{real\cdot pp}]{e,k} (\vec{\ell'}, \mathcal{B}^{\mathsf{p}}_{m+1})}$$

FIGURE 4.18 – Définition de la relation de transition $\xrightarrow[\mathsf{real\cdot pp}]{e,k}$

Exemple 4.4.1 – Le MSC μ_1 de la Figure 4.19.a est divisé en deux k-échanges. À la fin du k-échange e_1, nous avons $\mathcal{B}^{\mathsf{p}\prime}(r) = \{q\}$. q est ajouté à l'envoi de m_1. Pour le k-échange e_2, le message m_3 peut être reçu sans problème car $p \notin \mathcal{B}^{\mathsf{p}}(r)$ donc μ_1 est pp-réalisable.

Le MSC de la Figure 4.19.b est également divisé en k-échanges. À la fin du k-échange e_1, nous avons $\mathcal{B}^{\mathsf{p}\prime}(s) = \{r, p\}$. Dans le k-échange e_2, le message m_4 est envoyé par p à s et $p \in \mathcal{B}^{\mathsf{p}}(s)$ donc μ_2 n'est pas pp-réalisable.

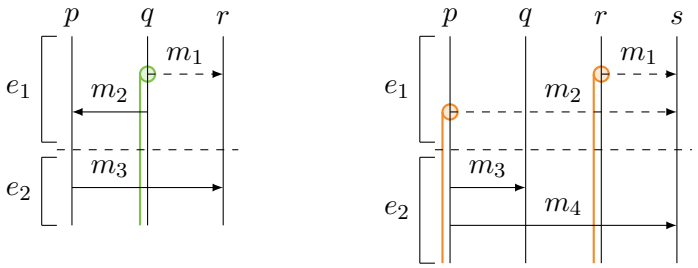


FIGURE 4.19 – MSC μ_1 l'évolution de l'ensemble $\mathcal{B}^{\mathsf{p}}(r)$ (en vert) (a) , et μ_2 avec l'évolution de l'ensemble $\mathcal{B}^{\mathsf{p}}(s)$ (en orange) (b)

Lemme 4.4.2. *Un MSC est k-synchrone si et seulement s'il existe une linéarisation $e = e_1 \cdots e_n$, un certain état global $\vec{\ell}$ et $\mathcal{B}^{\mathsf{p}} : \mathbb{P} \to 2^{\mathbb{P}}$ tels que*

$$(\vec{\ell_0}, \mathcal{B}^{\mathsf{p}}_{\emptyset}) \xrightarrow[\mathsf{real\cdot pp}]{e_1,k} \cdots \xrightarrow[\mathsf{real\cdot pp}]{e_n,k} (\vec{\ell}, \mathcal{B}^{\mathsf{p}})$$

Démonstration.

$\Rightarrow$ Soit μ un MSC k-synchrone. Alors $\exists e = e_1 \cdots e_n$ telle que e est une linéarisation de μ. La preuve marche par récurrence sur n.

Base Si $n = 1$ alors $e = e_1$. Alors il n'y a qu'un k-échange. Par hypothèse de récurrence, comme μ est pp-réalisable , on sait qu'il existe $\vec{\ell}$ et $\mathcal{B}^{\mathsf{p}}$ tels que $(\vec{\ell}, \mathcal{B}^{\mathsf{p}}_{\emptyset}) \overset{e}{\Rightarrow} (\vec{\ell}, \mathcal{B}^{\mathsf{p}})$. Par contradiction, supposons que $\exists \mathbf{m} = \{s_i, r_{i'}\}$ tel que $s_i = \mathtt{s}(p, q, m)$ et $p \in \mathcal{B}^{\mathsf{p}}_i(q)$. Alors, $\exists \mathbf{m}' = \{s_j\}$ tel que $s_j = \mathtt{s}(p, q, \mathbf{m}')$. Comme e est une linéarisation de μ et $\mathsf{proc}_{\mathtt{S}}(\mathbf{m}) = \mathsf{proc}_{\mathtt{S}}(\mathbf{m}')$, alors $j \prec_{po} i$. Comme $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}') = \mathsf{proc}_{\mathtt{R}}(\mathbf{m})$ et $\mathbf{m}$ est couplé alors que $\mathbf{m}'$ est non couplé, μ n'est pas pp-réalisable et donc nous arrivons à une contradiction.

Récurrence Si $n > 1$, par hypothèse de récurrence, nous avons

$$(\vec{\ell_0}, \mathcal{B}^{\mathsf{p}}_{\emptyset}) \xrightarrow[\mathsf{real\cdot pp}]{e_1,k} \cdots \xrightarrow[\mathsf{real\cdot pp}]{e_{n-1},k} (\vec{\ell}_{n-1}, \mathcal{B}^{\mathsf{p}})$$

Puisque toutes réceptions sont couplées à un envoi correspondant dans le k-échange courant, et que tout envois précèdent toutes les réceptions, nous avons

$$(\vec{\ell}_{n-1}, \mathcal{B}^{\mathsf{p}}_{\emptyset}) \overset{e}{\Rightarrow} (\vec{\ell}_{n}, \mathcal{B}^{\mathsf{p}})$$

pour un certain $\mathcal{B}^{\mathsf{p}}$.

L'hypothèse de récurrence implique que

$$\mathcal{B}^{\mathsf{p}}(q) = \{\mathsf{proc}_{\mathtt{S}}(\mathbf{m}) \mid \mathbf{m} \text{ est non matché } \& \ \mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = q\}$$

Par contradiction, on suppose que $\exists \mathbf{m} = \{s_i, r_{i'}\}$ tel que $s_i = \mathtt{s}(p, q, m)$ et $p \in \mathcal{B}^{\mathsf{p}}_i(q)$. Alors, il existe un message $\mathbf{m}'$ dans e tel que $\mathbf{m}' = \{s_j\}$, $s_j = \mathtt{s}(p, q, m')$. Comme $\mathbf{m}'$ dans e, et comme dans le cas de base, $j \prec_{po} i$ avec $\mathbf{m}$ couplé et $\mathbf{m}'$ non couplé, μ ne pourrait pas être pp-réalisable ce qui nous mène donc à une contradiction.

$\Leftarrow$ Si $e = e_1 \cdots e_n$ où chaque e_i correspond à un k-échange pp-réalisable. Montrons que $msc(e)$ est k-synchrone. Supponsons par contradiction que $msc(e)$ n'est pas k-synchrone. Comme e est une linéarisation de $msc(e)$ et est divisible en k-échanges pp-réalisables et que $msc(e)$ n'est pas pp-réalisable, alors, il existe $s_i = \mathtt{s}(p, q, m)$, $s_j = \mathtt{s}(p, q, m')$ tels que $i \prec j$ et :

- soit il existe $r_{i'} = \mathtt{r}(p, q, m)$, $r_{j'} = \mathtt{r}(p, q, m')$ tels que $j' \prec i'$ ou,
- soit s_i est non couplé et s_j est couplé.

De plus, nous avons que $s_i, s_j \in e_l$ ou $s_i \in e_l$ et $s_j \in e_m$ avec $l \neq m$. Quatre cas sont alors possibles.

- Dans le premier cas, s_i et s'_j sont couplés et appartiennent au même k-échange e_l. Alors, il n'y pas d'exécution telle que $(\vec{\ell}, \vec{c}_0) \stackrel{e_l}{\Longrightarrow} (\vec{\ell}', \vec{c})$ et e_l ne décrit pas un k-échange pp-réalisable.
- Dans le deuxième cas, s_i et s_j sont couplés mais n'appartiennent pas au même k-échange. Ce cas ne peut avoir lieu car $j' \prec i'$ implique que s_i et s_j doivent appartenir au même k-échange ou la réception du message s_i serait séparé de son envoi.
- Dans le troisième cas, s_i est non couplé et s_j est couplé, et ils apparaissent dans le même k-échange. Si s_i est non couplé, alors $p \in \mathcal{B}^{\mathsf{p}}_{i+1}(q)$. De plus, comme $i \prec j$, $p \in \mathcal{B}^{\mathsf{p}}_j(q)$ alors nous pouvons conclure que e_l n'est pas pp-réalisable.
- Dans le dernier cas, $s_i \in e_l$ est non couplé et $s_j \in e_m$ est couplé. Ainsi, comme la fonction $\mathcal{B}^{\mathsf{p}}$ est incrémentale, $p \in \mathcal{B}^{\mathsf{p}}(q)$ au début de e_m. Donc, $p \in \mathcal{B}^{\mathsf{p}}_j(q)$ et e_m n'est pas pp-réalisable, ce qui conclut cette preuve.

□

Nous pouvons alors conclure que, comme pour les sytèmes en boîte aux lettres, l'accessibilité est décidable pour les sytèmes communiquant en pair à pair.

Théorème 4.4.3. *Soit $\mathcal{S}$ un système k-synchronisable communiquant en pair à pair et $\vec{\ell}$ un état global de $\mathcal{S}$. Savoir si $\vec{\ell}$ est accessible dans $\mathcal{S}$ est décidable.*

Démonstration.

Il existe un nombre fini de configurations abstraites de la forme $(\vec{\ell}, \mathcal{B}^{\mathsf{p}})$ avec $\vec{\ell}$ un état de contrôle global et $\mathcal{B}^{\mathsf{p}} : \mathbb{P} \to (2^{\mathbb{P}})$. Ainsi, $\xrightarrow[\mathsf{real\cdot pp}]{e,k}$ est une relation sur un ensemble fini, et l'ensemble $\mathrm{MSC}_k(\mathcal{S})$ de MSC k-synchrone du système $\mathcal{S}$ forme un langage régulier. On peut déduire alors qu'il est décidable de savoir si une configuration abstraite donnée de la forme $(\vec{\ell}, \mathcal{B}^{\mathsf{p}})$ est accessible depuis la configuration initiale du système suivant une exécution k-synchronisable qui sera une linéarisation d'un MSC contenu dans $\mathrm{MSC}_k(\mathcal{S})$.

□

4.4.3 La décidabilité de la k-synchronisabilité

Comme pour les systèmes communiquant en boîte aux lettres, la détection d'une exécution critique détermine si un système est k-synchronisable ou non. Nous sommes donc à la recherche d'une exécution faisable et mauvaise. Le Lemme 4.3.1, dont nous remettons l'énoncé, est toujours valable ici.

Lemme 4.3.1. *Un système $\mathcal{S}$ n'est pas k-synchronisable si et seulement s'il existe une exécution e' de $\mathcal{S}'$ qui est faisable et mauvaise.*

La relation de transition $\xrightarrow[\mathsf{feas\cdot pp}]{e,k}$, en Figure 4.20, permet d'obtenir les exécutions faisables.

Contrairement à la sémantique boîte aux lettres, nous n'avons plus seulement besoin de conserver le destinataire du message dévié dest_π, mais également l'expéditeur exp_π. La règle de transition vérifie alors qu'il n'y a pas de message empêchant d'être pp-réalisable, c'est-à-dire, aucun message couplé envoyé de exp_π à dest_π après la déviation. Les ensembles de $\mathcal{B}_\pi$ ne sont alors plus utiles, car ils permettaient de retracer le chemin causal entre les deux actions.

$$\frac{(\vec{\ell},\mathcal{B}^{\mathsf{p}}) \xRightarrow[\mathsf{real\cdot pp}]{e,k} (\vec{\ell'},\mathcal{B}^{\mathsf{p}\prime}) \qquad e=a_1\cdots a_n \qquad (\forall \mathbf{m})\mathsf{proc}_{\mathtt{S}}(\mathbf{m})\neq\pi \quad (\forall \mathbf{m},\mathbf{m}')\mathsf{proc}_{\mathtt{R}}(\mathbf{m})=\mathsf{proc}_{\mathtt{R}}(\mathbf{m}')=\pi \implies \mathbf{m}=\mathbf{m}'\wedge\mathsf{dest}_\pi=\perp \quad (\forall i)a_i=\mathtt{s}(p,\pi,(q,m)) \implies \mathsf{dest}'_\pi=q\wedge\mathsf{exp}'_\pi=p\wedge d=i \quad \mathsf{dest}_\pi\neq\perp \implies \mathsf{dest}'_\pi=\mathsf{dest}_\pi \quad \mathsf{dest}'_\pi\neq\perp\wedge\mathsf{dest}_\pi=\perp \implies \nexists\mathbf{m}(\mathbf{m}\text{ est couplé }\wedge s_j\in\mathbf{m}\wedge j>d \wedge\mathsf{proc}_{\mathtt{S}}(\mathbf{m})=\mathsf{exp}'_\pi\wedge\mathsf{proc}_{\mathtt{R}}(\mathbf{m})=\mathsf{dest}'_\pi) \quad \mathsf{dest}'_\pi\neq\perp\wedge\mathsf{dest}_\pi\neq\perp \implies \nexists\mathbf{m}(\mathbf{m}\text{ est couplé }\wedge\mathsf{proc}_{\mathtt{S}}(\mathbf{m})=\mathsf{exp}_\pi\wedge\mathsf{proc}_{\mathtt{R}}(\mathbf{m})=\mathsf{dest}_\pi)}{(\vec{\ell},\mathcal{B}^{\mathsf{p}},\mathsf{exp}_\pi,\mathsf{dest}_\pi) \xRightarrow[\mathsf{feas\cdot pp}]{e,k} (\vec{\ell}\,',\mathcal{B}^{\mathsf{p}\prime},\mathsf{exp}'_\pi,\mathsf{dest}'_\pi)}$$

FIGURE 4.20 – Définition de la relation de transition $\xRightarrow[\mathsf{feas\cdot pp}]{e,k}$

Lemme 4.4.4. *Soit e' une exécution de $\mathcal{S}'$. Alors e' est une exécution k-synchronisable et faisable si et seulement s'il existe $e''=e_1\cdots e_n\cdot\mathtt{s}(\pi,q,m)\cdot\mathtt{r}(\pi,q,m)$ telle que*

$$(\vec{\ell_0},\mathcal{B}^{\mathsf{p}}_{\emptyset},\perp,\perp)\xRightarrow[\mathsf{feas\cdot pp}]{e_1,k}\ldots\xRightarrow[\mathsf{feas\cdot pp}]{e_n,k}(\vec{\ell},\mathcal{B}^{\mathsf{p}\prime},p,q).$$

avec $e_1,\ldots,e_n\in\mathtt{S}^{\leq k}\mathtt{R}^{\leq k}$, $msc(e')=msc(e'')$, $\mathcal{B}^{\mathsf{p}\prime}:\mathbb{P}\to 2^{\mathbb{P}}$, un état global $\vec{\ell}$ et $p,q\in\mathbb{P}$.

Démonstration.

$\Rightarrow$ Soit e' une exécution k-synchronisable et faisable de $\mathcal{S}'$. On montre qu'il existe

$$e''=e_1\cdots e_n\cdot\mathtt{s}(\pi,q,m)\cdot\mathtt{r}(\pi,q,m)$$

avec $e_1,\ldots,e_n\in\mathtt{S}^{\leq k}\mathtt{R}^{\leq k}$, tel que $msc(e')=msc(e'')$, $\mathcal{B}^{\mathsf{p}\prime}:\mathbb{P}\to 2^{\mathbb{P}}$, un état global $\vec{\ell}\,'$ et des processus p et q tels que

$$(\vec{\ell_0},\mathcal{B}^{\mathsf{p}}_{\emptyset},\perp,\perp)\xRightarrow[\mathsf{feas\cdot pp}]{e_1,k}\ldots\xRightarrow[\mathsf{feas\cdot pp}]{e_n,k}(\vec{\ell}\,',\mathcal{B}^{\mathsf{p}\prime},p,q).$$

Puisque e' est k-synchronisable, $msc(e')$ est k-synchrone et il existe e'' tel que $msc(e'')=msc(e')$ et $e''=e_1\cdots e_n$ où chaque e_i est un k-échange pp-réalisable. Ainsi, il existe $\vec{\ell}\,'$ et $\mathcal{B}^{\mathsf{p}\prime}$ tel que $(\vec{\ell},\mathcal{B}^{\mathsf{p}})\xRightarrow[\mathsf{real\cdot pp}]{e_1,k}\cdots\xRightarrow[\mathsf{real\cdot pp}]{e_n,k}(\vec{\ell}\,',\mathcal{B}^{\mathsf{p}\prime})$.

Puisque e' est faisable, posons $e\cdot r\in\mathrm{EX}(\mathcal{S})$ tel que $\mathsf{deviate}(e\cdot r)=e'$. Donc il existe un envoi au processus π dans e' et $e'=e'_1\cdot\mathtt{s}(p,\pi,(q,m))\cdot\mathtt{r}(p,\pi,(q,m))\cdot e'_2\cdot\mathtt{s}(\pi,q,m)\cdot\mathtt{r}(\pi,q,m)$. Alors il existe un unique envoi à π tel que $\mathsf{exp}_\pi=p$ and $\mathsf{dest}_\pi=q$.

Par contradiction, supposons qu'il existe un message couplé $\mathbf{m}'=\{a_i,a_j\}$ appartenant à un k-échange dans e'_2 tel que $\mathsf{proc}_{\mathtt{S}}(\mathbf{m}')=p$ et $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}')=q$. Maintenant, si on considère la séquence non-déviée $e\cdot r$, soit $a_{i'}=\mathtt{s}(p,q,m)$ et $r=a_{j'}=\mathtt{r}(p,q,m)$. Nous avons donc $i'\prec i$ et $j\prec j'$ empêchant d'être pp-réalisable et le fait que $e\cdot r$ est une exécution.

$\Leftarrow$ Prenons une séquence d'action e'' telle que

$$(\vec{\ell_0},\mathcal{B}^{\mathsf{p}}_{\emptyset},\perp,\perp)\xRightarrow[\mathsf{feas\cdot pp}]{e_1,k}\cdots\xRightarrow[\mathsf{feas\cdot pp}]{e_n,k}(\vec{\ell}\,',\mathcal{B}^{\mathsf{p}\prime},p,q)$$

et e' une exécution de $\mathcal{S}'$ telle que $msc(e'') = msc(e')$.

Par contradiction, supposons que e' n'est pas faisable. Alors, $e' = \mathsf{deviate}(e \cdot r)$ avec $e \cdot r$ qui n'est pas une exécution de $\mathcal{S}$ car elle n'est pas pp-réalisable. Si $e \cdot r$ n'est pas pp-réalisable, mais que e' oui, alors $\exists \mathbf{m} = \{s_i, r_{i'}\}, \mathbf{m}' = \{s_j, r_{j'}\}$ tels que $s_i = \mathtt{s}(p,q,m), r_{i'} = \mathtt{r}(p,q,m)$ et $s_j = \mathtt{s}(p,q,m'), r_{j'} = \mathtt{r}(p,q,m')$ et dans $msc(e \cdot r) : i \prec j$ et $j' \prec i'$. Dans e', il existe une action $s_l = \mathtt{s}(p,\pi,(q,m))$ telle que dans $msc(e') : l \prec j$. De plus, l'envoi de π étant la dernière action, posons $\mathtt{r}(p,q,m') = r_m$ alors $j' \prec m$. Alors, il exsite un k-échange après la déviation du message m où le message m' apparaît. Ainsi, nous avons $\mathsf{dest}'_\pi \neq \bot$, $\mathsf{proc}_\mathtt{S}(\mathbf{m}') = p$ et $\mathsf{proc}_\mathtt{R}(\mathbf{m}') = q$. De plus, si s_l appartient au k-échange courant alors $l < j$. Ceci implique que la transition $\xrightarrow[\mathsf{feas\cdot pp}]{e',k}$ n'est pas valide, ce qui nous amène à une contradiction. Donc e' doit être une exécution k-synchronisable et faisable du système $\mathcal{S}'$.

□

Enfin, la méthode de reconnaissance des mauvaises exécutions est identique à celle décrite pour les systèmes en boîte aux lettres. La relation de transition utilisée est alors $\xrightarrow[\mathsf{bad}]{e,k}$. Le Lemme 4.3.4 est valable quelque soit la communication et ainsi, le Lemme 4.3.5 est valide avec cette communication également. Nous les rappelons.

Lemme 4.3.4. *Une exécution faisable e' est mauvaise si et seulement si au moins l'une des deux affirmations suivantes est vraie :*

1. $\mathbf{m}_{\mathsf{start}} \rightarrow^* \xrightarrow{\mathrm{RS}} \rightarrow^* \mathbf{m}_{\mathsf{stop}}$
2. *l'ensemble* $\mathsf{Post}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{Pre}^*(\mathbf{m}_{\mathsf{stop}})$ *est de taille supérieure ou égale à* $k+2$.

Lemme 4.3.5. *Soit e' une exécution faisable et k-synchronisable de $\mathcal{S}'$. Alors, e' est mauvaise si et seulement s'il existe $e'' = e_1 \cdots e_n \cdot \mathtt{s}(\pi,q,m) \cdot \mathtt{r}(\pi,q,m)$ telle que*

$$(\{\pi\}, Q, 0, \mathsf{False}, \mathtt{lastisRec}_0) \xrightarrow[\mathsf{bad}]{e_1,k} \cdots \xrightarrow[\mathsf{bad}]{e_n,k} (P', \{\pi\}, \mathtt{cnt}, \mathtt{sawRS}, \mathtt{lastisRec})$$

avec une des deux affirmations suivantes vraie :

$\mathtt{sawRS} = \mathsf{True}$ *ou* $\mathtt{cnt} = k+2$

et $e_1, \cdots, e_n \in \mathtt{S}^{\leq k}\mathtt{R}^{\leq k}$ *et* $msc(e') = msc(e''), P', Q \subseteq \mathbb{P}, \mathtt{sawRS} \in \{\mathsf{True}, \mathsf{False}\}, \mathtt{cnt} \in \{0, \cdots, k+2\}$.

Nous pouvons alors conclure que, pour un k donné, la k-synchronisabilité est décidable.

Théorème 4.4.5. *La k-synchronisabilité d'un système $\mathcal{S}$ communiquant en pair à pair est décidable pour un k donné.*

Démonstration.

Soit $\mathcal{S}$ un système donné. Par les Lemmes 4.3.1, 4.4.4 et 4.3.5, $\mathcal{S}$ n'est pas k-synchronisable si et seulement s'il existe une séquence d'actions $e' = e'_1 \cdots e'_n \cdot s \cdot r$ telle que $e_i \in \mathtt{S}^{\leq k}\mathtt{R}^{\leq k}$, $s = \mathtt{s}(\pi,q,m), r = \mathtt{r}(\pi,q,m)$,

$$(\vec{\ell_0}, \mathcal{B}^{\mathsf{p}}_{\emptyset}, \bot, \bot) \xrightarrow[\mathsf{real\cdot pp}]{e'_1,k} \ldots \xrightarrow[\mathsf{real\cdot pp}]{e'_n,k} (\vec{\ell}', \mathcal{B}^{\mathsf{p}\prime}, \mathsf{exp}_\pi, q)$$

et

$$(\{\pi\}, Q, \mathsf{False}, 0) \xRightarrow[\mathsf{bad}]{e'_1,k} \dots \xRightarrow[\mathsf{bad}]{e'_n,k} \xRightarrow[\mathsf{bad}]{s\cdot r,k} (P', \{\pi\}, \texttt{sawRS}, \texttt{cnt})$$

pour certains $\vec{\ell}', \mathcal{B}^{\mathsf{p}\prime}, \mathsf{exp}_\pi, Q, P'$ avec $\mathsf{exp}_\pi \notin \mathcal{B}^{\mathsf{p}}(\mathsf{dest}_\pi)$. Puisque les relations $\xRightarrow[\mathsf{real}\cdot\mathsf{pp}]{e,k}$ et $\xRightarrow[\mathsf{bad}]{e,k}$ sont sur un nombre fini d'états, l'existence d'une telle séquence est décidable.

□

4.5 Conclusion

Ce chapitre présentait en détail les travaux publiés dans [Di Giusto et al., 2020]. Ceux-ci consistent en la correction des preuves présentées dans [Bouajjani et al., 2018a]. On y prouve alors la décidabilité de l'accessibilité dans un système k-synchronisable communicant en boîte aux lettres, grâce à une caractérisation graphique des MSC mb-réalisables d'une part et des MSC k-synchrones d'autre part. On prouve également qu'il est décidable de savoir si un système donné communiquant en boîte aux lettres est k-synchronisable pour un k donné. Ces preuves sont accompagnées de comparaisons précises avec celles présentées dans [Bouajjani et al., 2018a]. La dernière section consiste en l'adaptation de nos preuves aux systèmes communiquant en pair à pair. Ainsi, l'accessibilité dans un système k-synchronisable et l'appartenance à la classe des systèmes k-synchronisables pour un k donné sont décidables dans les deux types de communication considérés dans cette thèse.

CHAPITRE 5

Deviner le k d'un système k-synchronisable

Nous avons vu dans le chapitre précédent que l'on peut décider si un système est k-synchronisable pour un k donné. Cela nous a amenés à nous poser une nouvelle question, jusqu'alors sans réponse : saurait-on trouver un k tel que le système donné puisse être k-synchronisable ? Dans [Bouajjani et al., 2018b], les auteurs répondent à ce problème mais uniquement pour un sous-ensemble de systèmes nommés *flow-bounded*. Nous verrons qu'il est en vérité décidable pour tous les systèmes communiquant en boîte aux lettres et c'est ce que nous prouverons dans ce chapitre. Le même problème pour les systèmes communiquant en pair à pair reste ouvert.

Le chapitre se décompose comme suit. On constatera dans une première section que le k que nous cherchons correspond à la taille d'un k-échange particulier : le plus grand échange accessible et premier. En effet, la procédure consistera à construire tous les k-échanges que l'on peut trouver dans le système. Il faut s'assurer alors qu'ils appartiennent à une exécution k-synchronisable, c'est ce qu'on appellera être *accessible*. D'autre part, si un k-échange peut être divisé en deux k-échanges plus petits, il ne doit pas être considéré car l'on cherche le plus petit k tel que le système est k-synchronisable. On s'assurera que les k-échanges ne soient pas divisibles en k-échanges plus petits et c'est ce qu'on appellera être *premier*. Comme on cherche une borne sur la taille des k-échanges, notre but sera donc de trouver le plus grand des k-échanges accessibles et premiers. Il s'avère que si k est sa taille, alors soit le système observé est k-synchronisable, soit le système n'est pas k-synchronisable pour aucun k.

Ainsi, pour parvenir à calculer ce k, il s'agira alors de construire les k-échanges accessibles dans la deuxième section et les k-échanges premiers dans la troisième. La dernière section regroupera ces ensembles pour procéder au calcul final.

5.1 Le plus grand échange accessible et premier

Cette section a pour but de démontrer le fondement de la recherche du plus grand échange accessible et premier. Définissons d'abord formellement cette notion.

Définition 5.1.1 (Échange, accessible, premier)**.** On définit :

- ✢ Un *échange* est un k-échange pour un certain k arbitraire, et on appelle k la taille de cet échange,

- Un échange μ est *accessible* s'il existe une séquence d'échanges $\mu_1, \cdots, \mu_n$ pour un certain $n \geq 0$ et telle que $\mu_1 \cdots \mu_n \cdot \mu \in \text{MSC}(\mathcal{S})$,
- Un échange μ est *premier* s'il n'existe pas de décomposition $\mu = \mu_1 \cdot \mu_2$ en deux échanges non vides.

Exemple 5.1.1 – Par exemple, le MSC μ en Figure 5.1 est un 2-échange qui a comme linéarisation possible :

$$\mathtt{s}(p, q, m_1) \cdot \mathtt{s}(r, q, m_2) \cdot \mathtt{r}(p, q, m_1) \cdot \mathtt{r}(r, q, m_2)$$

Il n'est pas premier, car il peut être factorisé en 1-échanges, comme suit :

$$\mathtt{s}(p, q, m_1) \cdot \mathtt{r}(p, q, m_1) \ \cdot \ \mathtt{s}(r, q, m_2) \cdot \mathtt{r}(r, q, m_2)$$

On constate que cette division est en effet possible, on la retrouve en pointillés sur la figure.

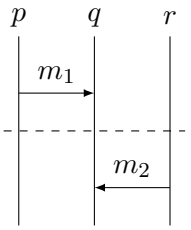


FIGURE 5.1 – MSC μ de l'Exemple 5.1.1

Pour un système k-synchronisable, tous les k-échanges sont bornés par k. Toutefois, tous les systèmes ne sont pas k-synchronisables. Quand c'est le cas, on peut estimer que la borne des k-échanges présents dans le système est l'infini.

On appellera cette borne le *degré de synchronisabilité* du système.

Définition 5.1.2 (Degré de synchronisabilité)**.** Le degré de synchronisabilité $\mathsf{sd}(\mathcal{S})$ d'un système $\mathcal{S}$ est le plus petit k tel que $\mathcal{S}$ est k-synchronisable. En particulier, $\mathsf{sd}(\mathcal{S}) = \infty$ si $\mathcal{S}$ n'est pas k-synchronisable pour aucun k.

Nous pouvons alors établir le lien entre le degré de synchronisabilité et le plus grand échange accessible et premier.

Lemme 5.1.1. *Soit $k \in \mathbb{N} \cup \{\infty\}$ le maximum des tailles de tous les échanges accessibles et premiers.*

1. *Si $k = \infty$, alors $\mathsf{sd}(\mathcal{S}) = \infty$;*
2. *Si $k < \infty$, alors soit $\mathcal{S}$ est k-synchronisable et $\mathsf{sd}(\mathcal{S}) = k$, soit $\mathcal{S}$ n'est pas k-synchronisable et $\mathsf{sd}(\mathcal{S}) = \infty$.*

Démonstration.

Soit $k \in \mathbb{N} \cup \{\infty\}$ le maximum des tailles de tous les échanges accessibles et premiers. Supposons qu'il existe K tel que $\mathcal{S}$ est K-synchronisable. Montrons que $k \leq K$ et que $\mathcal{S}$ est k-synchronisable.

- $k \leq K$. Supposons par contradiction que $k \geq K+1$. Alors il existe une séquence $\mu_1, \cdots, \mu_n$ telle que $\mu_1 \cdots \mu_n \cdot \mu \in \text{MSC}(\mathcal{S})$ avec μ premier de taille $K+1$. Puisque μ est premier, il correspond à une composante fortement connexe de taille $K+1$ du graphe de conflits de $\mu_1 \cdots \mu_n \cdot \mu$. Alors, par le Lemme 4.1.2, $\mu_1 \cdots \mu_n \cdot \mu$ ne peut pas être K-synchrone, ce qui est donc une contradiction.
- $\mathcal{S}$ est k-synchronisable. Fixons $\mu \in \text{MSC}(\mathcal{S})$ et montrons qu'il peut être divisé en séquence de k-échanges. Par hypothèse, $\mathcal{S}$ est K-synchronisable, donc il existe $\mu_1, \cdots, \mu_n$ tels que $\mu = \mu_1 \cdots \mu_n$. Supposons que cette séquence fut préalablement décomposée en échanges premiers, alors on peut affirmer que chaque μ_i est premier. De plus, ils sont tous accessibles, donc leur taille est bornée par k. Ainsi, μ peut être divisé en séquence de k-échanges.

□

On a vu dans la section précédente que savoir si un système est k-synchronisable pour un k donné est décidable. L'objectif est alors de calculer la taille k du plus grand échange accessible et premier pour obtenir le degré de synchronisabilité $\mathtt{sd}(\mathcal{S})$.

Pour calculer k, deux problèmes restent à résoudre : le nombre d'échanges est possiblement infini et l'on devrait examiner des séquences d'échanges arbitrairement longues. Pour y remédier, nous réduisons cela à un problème sur des langages réguliers.

Pour faciliter la lecture, nous écrirons les envois et les réceptions sous une nouvelle forme. Nous définissons un nouvel alphabet $\Sigma = \{!?, !\} \times \mathbb{V} \times \mathbb{P}^2$. Nous noterons $!?m^{p\to q}$ pour l'envoi et la réception d'un message m envoyé de p vers q (et respectivement $!m^{p\to q}$ si m n'est pas couplé) pour un mot de Σ. Tout mot $w \in \Sigma^*$ est associé à un MSC $msc(w)$. Considérons ces deux substitutions :

$$\sigma_1 : \Sigma \to \mathtt{S} \text{ telle que } \quad \sigma_1(!?m^{p\to q}) = \sigma_1(!m^{p\to q}) = \mathtt{s}(p,q,m)$$

$$\sigma_2 : \Sigma \to \mathtt{R} \text{ telle que } \begin{cases} \sigma_2(!?m^{p\to q}) = \mathtt{r}(p,q,m) \\ \sigma_2(!m^{p\to q}) = \varepsilon \end{cases}$$

Alors le MSC associé à un mot w est définit par $msc(w) = msc(\sigma_1(w)\sigma_2(w))$. Par construction, ceci est alors un échange puisqu'il admet une linéarisation dans $\mathtt{S}^* \cdot \mathtt{R}^*$. De plus, tout échange accessible est alors représentable par un tel mot.

Lemme 5.1.2. *Pour tout échange accessible μ, il existe $w \in \Sigma^*$ tel que $\mu = msc(w)$.*

Démonstration.

Soient μ un échange accessible, et $\mu_1, \cdots, \mu_n$ une séquence telle que $\mu_1 \cdots \mu_n \cdot \mu \in \text{MSC}(\mathcal{S})$. Alors, il existe une linéarisation lin_μ telle que $msc(lin_\mu) = \mu_1 \cdots \mu_n$ et telle que $lin_\mu \in \text{EX}(\mathcal{S})$ et respecte donc la sémantique boîte aux lettres. Alors, lin_μ définit une énumération des envois de μ $\mathtt{s}(p_1,q_1,m_1), \cdots, \mathtt{s}(p_n,q_n,m_n)$. Soit $w = a_1 \cdots a_n$ où a_i est soit $!?m_i^{p_i\to q_i}$ si $\mathtt{s}(p_i,q_i,m_i)$ est couplé dans μ, soit $!m_i^{p_i\to q_i}$ s'il est non couplé.

On cherche alors à prouver que $msc(w) = \mu$, ou, autrement dit, $\sigma_1(w)\sigma_2(w)$ est une linéarisation de μ. Par contradiction, supposons que ce ne soit pas le cas. Alors il existe deux évènements e, e' tels que $e < e'$ dans l'énumération de $\sigma_1(w)\sigma_2(w)$ mais $(e', e) \in (\prec_{po} \cup \prec_{src})^*$:

- si e, e' sont deux envois, alors e se produit avant e' dans $\sigma_1(w)$, et donc dans lin_μ, ce qui est contradictoire avec $(e', e) \in (\prec_{po} \cup \prec_{src})^*$;

- si e est un envoi et e' une réception, alors $(e', e) \in (\prec_{po} \cup \prec_{src})^*$ contredit le fait que μ est un échange ;
- si e est une réception et e' un envoi, alors $e < e'$ est impossible au vu de la définition de σ_1 et σ_2.
- enfin, supposons que e et e' sont des réceptions. Comme $(e', e) \in (\prec_{po} \cup \prec_{src})^*$, on peut déduire que $e' \prec_{po} e$ parce que μ est un échange. Soit s, s' les envois couplés respectivement à e, e'. Comme $e < e'$ dans $\sigma_1(w)\sigma_2(w)$, $s < s'$ dans $\sigma_1(w)\sigma_2(w)$, et donc $s < s'$ dans lin_μ. Cependant, $e' < e$ dans lin_μ car $e' \prec_{po} e$, ce qui est impossible dans la sémantique boîte aux lettres et mène donc à une contradiction.

□

La preuve de ce lemme découle du fait qu'il est toujours possible de recevoir des messages dans le même ordre que celui dans lequel ils ont été envoyés. Cette déduction peut être faite grâce à la communication en boîte aux lettres. Cette propriété n'est d'ailleurs pas correcte dans une communication en pair à pair. Voici un exemple en ce sens.

Exemple 5.1.2 – Considérons le MSC μ de Figure 5.2. Ce MSC n'est pas mb-réalisable. En effet, l'envoi de m_1 doit se produire avant l'envoi de m_4 mais la réception de m_4 doit avoir lieu avant la réception de m_1. Pour cette raison, il n'existe pas de mot w tel que $msc(w)$ correspondrait à cet MSC : un tel mot donnerait une linéarisation correspondant à une exécution mb-réalisable. D'un autre côté, cet MSC est pp-réalisable. Par exemple, la linéarisation e suivante correspond à une exécution correcte en pair à pair :

$$e = \mathtt{s}(p, q, m_3) \cdot \mathtt{s}(p, r, m_4) \cdot \mathtt{s}(s, r, m_1) \cdot \mathtt{s}(s, q, m_2) \cdot \\ \mathtt{r}(s, q, m_2) \cdot \mathtt{r}(p, q, m_3) \cdot \mathtt{r}(p, r, m_4) \cdot \mathtt{r}(s, r, m_1)$$

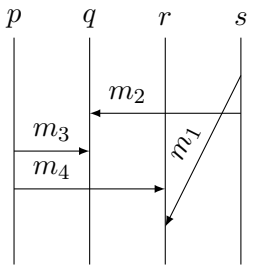


FIGURE 5.2 – Un MSC pas mb-réalisable et qu'on ne peut pas représenter avec un mot de Σ

Poursuivons alors avec les deux langages sur Σ correspondant à notre recherche : les échanges accessibles et les échanges premiers.

$$\mathcal{L}_a = \{w \in \Sigma^* \mid msc(w) \text{ est accessible } \}$$
$$\mathcal{L}_p = \{w \in \Sigma^* \mid msc(w) \text{ est premier } \}$$

Ainsi, la borne k que nous cherchons est donc la longueur du mot le plus long dans l'intersection $\mathcal{L}_a \cap \mathcal{L}_p$, où $\mathcal{L}_a$ est le langage des échanges accessibles, et $\mathcal{L}_p$ celui des échanges premiers. Il suffit alors de démontrer que $\mathcal{L}_a$ et $\mathcal{L}_p$ sont effectivement des langages réguliers pour obtenir un algorithme permettant de calculer k. C'est ce que nous ferons dans les deux prochaines sections.

5.2 Les échanges accessibles

Nous commencerons donc par les échanges accessibles. Dans cette section, nous définirons l'automate fini qui accepte un mot $w \in \Sigma^*$ si et seulement si $msc(w)$ est accessible. Rappelons qu'un MSC est accessible s'il existe une séquence de MSC $\mu_1, \cdots \mu_n$ telle que $\mu_1 \cdots \mu_n \cdot msc(w) \in \textsc{Msc}(\mathcal{S})$.

Le préfixe $\mu_1 \cdots \mu_n$ amène le système dans une configuration qui conditionne les actions qui peuvent être faites dans $msc(w)$ pour rester cohérent avec la sémantique en boîte aux lettres. Plus précisément, la présence d'un message non couplé dans un canal impose qu'aucun message envoyé après et dans le même canal ne peut être lu et donc être couplé.

La construction de l'automate acceptant le langage $\mathcal{L}_a$ se fait en 3 étapes.

- Les premiers automates que nous construisons reconnaissent chacun les mots qui encodent un échange pouvant commencer dans un état de contrôle donné, qui passeront pas un autre état de contrôle après les envois et qui finiront dans un état de contrôle final après les réceptions. On note alors $\overrightarrow{in}$ l'état initial, $\overrightarrow{mid}$ l'état de contrôle après les envois et $\overrightarrow{fin}$ l'état de contrôle final. Un automate dépendra d'un triplet d'états de contrôle comme celui-ci et reconnaîtra les séquences d'actions correspondantes ayant la forme d'un échange : envois puis réceptions.

- Nous traitons par la suite les MSC mb-réalisables. On définit des automates où les états sont des contenus de canaux abstraits et les transitions des mots de Σ^*. Comme dans le chapitre précédent, nous ne pouvons pas considérer le véritable état des canaux : il y aurait une infinité de possibilités. Alors, le même type d'abstraction que précédemment est utilisée, on se souvient seulement des processus dont les prochains messages envoyés ou reçus auraient un lien SS avec un message non couplé destiné à un processus p. On utilisera donc les ensembles $\mathcal{C}_{\mathtt{S},p}$ et $\mathcal{C}_{\mathtt{R},p}$ pour savoir si le message observé est réalisable compte tenu des actions passées. Comme vu également, les ensembles $\mathcal{C}_{\mathtt{R},p}$ nous indiqueront si la séquence de messages est mb-réalisable avec l'absence du processus p.

- Si l'on fait la combinaison d'un des premiers automates avec un automate du second type, on pourra construire un automate qui reconnaît les mots correspondant à un échange tout en vérifiant qu'il est mb-réalisable. Et enfin, si l'on met à la suite ces automates combinés, on peut obtenir, en commençant par la configuration initiale du système, les mots correspondants à des échanges mb-réalisables et accessibles et donc le langage $\mathcal{L}_a$.

Exemple 5.2.1 – On peut imaginer alors qu'un MSC peut être découpé en fonction des états de contrôle du système dans lequel le système se trouve au moment des actions. La Figure 5.3.a est un MSC tel qu'on a l'habitude de les représenter. La Figure 5.3.b est une réprésentation du même MSC avec les moments où les états de contrôle sont atteints : $\overrightarrow{in}$ est l'état initial du MSC, $\overrightarrow{mid}$ est atteint après tous les envois et $\overrightarrow{fin}$ après toutes les réceptions. On peut constater que l'état de l'automate q est le même dans $\overrightarrow{in}$ et dans $\overrightarrow{mid}$, tandis que l'automate p est dans le même état dans $\overrightarrow{mid}$ et dans $\overrightarrow{fin}$.

De plus, on retrouve aussi la représentation des ensembles de $\mathcal{B}$, en particulier $\mathcal{C}_{\mathtt{S},s}$ et $\mathcal{C}_{\mathtt{R},s}$, respectivement en orange et en vert. Ici, l'on constate que $\mathcal{B}$ n'était pas vide : $p \in \mathcal{C}_{\mathtt{S},s}$, il résulte à la fin du MSC que $\mathcal{C}'_{\mathtt{S},s} = \{p, r\}$ et $\mathcal{C}'_{\mathtt{R},s} = \{q, s\}$, et donc ce MSC n'est pas mb-réalisable avec un tel état initial pour les canaux.

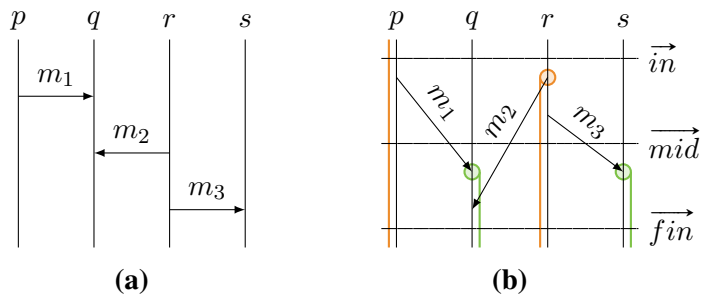


FIGURE 5.3 – Un MSC (a) et sa division en états de contrôle avec le contenu des ensembles $\mathcal{C}_{\mathtt{S},s}$ (en orange) et $\mathcal{C}_{\mathtt{R},s}$ (en vert) (b)

5.2.1 Des séquences d'actions avec la forme d'un échange

Cette première étape a pour but de construire les automates reconnaissant les mots encodants une séquence d'actions jouables dans le système ayant la forme d'un échange. On considère un triplet d'états de contrôle $((\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin}))$ tels que $\overrightarrow{in}$ est l'état de contrôle de départ, $\overrightarrow{mid}$ est l'état de contrôle atteint après tous les envois et $\overrightarrow{fin}$ est l'état de contrôle atteint après toutes les réceptions.

Les états de ces automates sont simplement les états de contrôle du système. Les transitions sont étiquetées par un message, couplé ou non, de la forme donc $!?m^{p\to q}$ ou $!m^{p\to q}$ (" !" représentant l'envoi et " ?" la réception).

Intuitivement, l'obtention d'un tel système se fait grâce à deux projections du système : l'une ne concervant que les transitions d'envois, l'autre seulement les transitions de réceptions. L'automate que nous construisons avance dans les deux projections à la fois. On autorise un envoi non couplé $(\overrightarrow{\ell_s}, \overrightarrow{\ell_r}) \xrightarrow{!m^{p\to q}} (\overrightarrow{\ell_s}', \overrightarrow{\ell_r})$ si l'envoi est disponible dans la projection depuis l'état $\overrightarrow{\ell_s}$. On remarque que l'état dans la projection des réceptions ne change pas puisqu'aucune réception n'est réalisée. On autorise un envoi couplé $(\overrightarrow{\ell_s}, \overrightarrow{\ell_r}) \xrightarrow{!?m^{p\to q}} (\overrightarrow{\ell_s}', \overrightarrow{\ell_r}')$ si l'envoi et la réception correspondante sont disponibles.

On rappelle que l'on ne regarde pas les canaux : appartenir à un de ces langages n'implique pas d'être mb-réalisable. Cela constituera notre prochaine étape.

Formellement, on définit ces automates comme suit :

Définition 5.2.1 (Automate SR). Soit $\mathcal{S}$ un système, $\overrightarrow{in}$, $\overrightarrow{mid}$ et $\overrightarrow{fin}$ des états de contrôle de $\mathcal{S}$. L'automate $\mathtt{SR}(\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin}) = (L_{\mathtt{SR}}, \delta_{\mathtt{SR}}, \ell^0_{\mathtt{SR}}, F_{\mathtt{SR}})$ est constitué de :

- $L_{\mathtt{SR}} = \{(\overrightarrow{\ell}, \overrightarrow{\ell}') \mid \overrightarrow{\ell}, \overrightarrow{\ell}' \in L_{\mathcal{S}}\}$, où, pour rappel, $L_{\mathcal{S}}$ désigne l'ensemble des états de contrôle du système $\mathcal{S}$;
- $\ell^0_{\mathtt{SR}} = (\overrightarrow{in}, \overrightarrow{mid})$;
- $F_{\mathtt{SR}} = \{(\overrightarrow{mid}, \overrightarrow{fin})\}$;
- pour toute transition $(\overrightarrow{\ell}_s, \mathtt{s}(p,q,m), \overrightarrow{\ell}'_s) \in \delta_{\mathcal{S}}$:
  - $((\overrightarrow{\ell}_s, \overrightarrow{\ell}), !m^{p\to q}, (\overrightarrow{\ell}'_s, \overrightarrow{\ell})) \in \delta_{\mathtt{SR}}$ pour $\overrightarrow{\ell} \in L_{\mathcal{S}}$;
  - s'il existe $(\overrightarrow{\ell}_r, \mathtt{r}(p,q,m), \overrightarrow{\ell}'_r) \in \delta_{\mathcal{S}}$ alors $((\overrightarrow{\ell}_s, \overrightarrow{\ell}_r), !?m^{p\to q}, (\overrightarrow{\ell}'_s, \overrightarrow{\ell}'_r)) \in \delta_{\mathtt{SR}}$.

On note alors $\mathcal{L}(\mathtt{SR}(\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin}))$ le langage de cet automate. Voici un exemple d'un tel automate obtenu pour un système donné et un certain triplet d'états de contrôle.

Exemple 5.2.2 – Soit $\mathcal{S}_1$ un système composé des automates p, q, et r de la Figure 5.4.

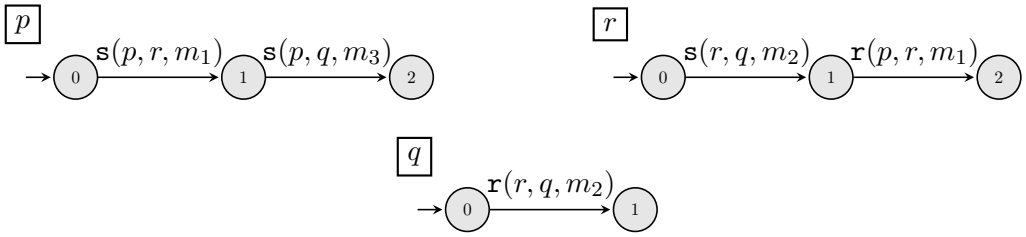


FIGURE 5.4 – Système $\mathcal{S}_1$

Pour le triplet $(\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin})$ où $\overrightarrow{in} = (0,0,0)$, $\overrightarrow{mid} = (2,0,1)$ et $\overrightarrow{fin} = (2,1,2)$, on obtient l'automate $\mathtt{SR}(\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin})$ de la Figure 5.5.

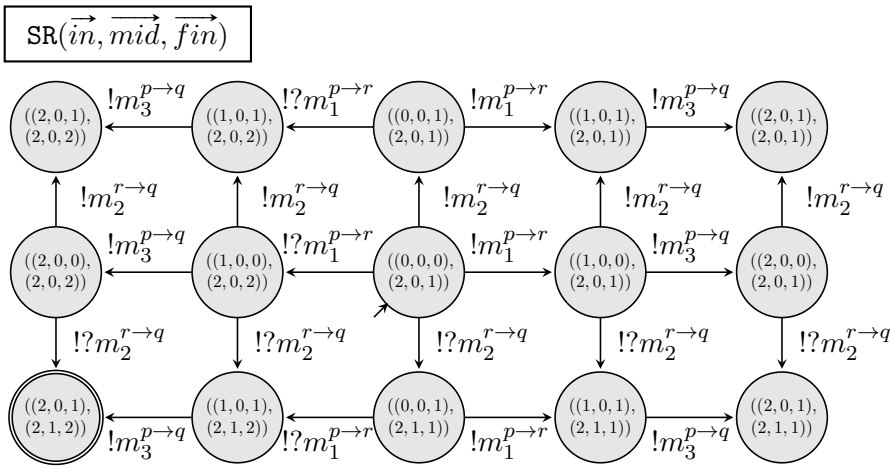


FIGURE 5.5 – Automate $\mathtt{SR}((0,0,0),(2,0,1),(2,1,2))$ pour le système $\mathcal{S}_1$

Le langage de cet automate est :

$$\mathcal{L}(\mathtt{SR}(\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin})) = !?m_1^{p\to r}(!m_3^{p\to q}!?m_2^{r\to q} + !?m_2^{r\to q}!m_3^{p\to q}) + !?m_2^{r\to q}!?m_1^{p\to r}!m_3^{p\to q}$$

Ce langage contient 3 mots. On représente les MSC de ces échanges encodés dans la Figure 5.6. Autrement dit, ce sont des échanges (s'ils sont mb-réalisables, ce qui est le cas ici) jouables depuis l'état $\overrightarrow{in}$. Leur séquence d'envois amène le système dans l'état $\overrightarrow{mid}$ et leurs réceptions dans l'état $\overrightarrow{fin}$. On peut aussi constater que les MSC des Figures 5.6.a et 5.6.b sont en réalité identiques : la différence visuelle est créée par l'ordre des réceptions des messages m_2 et m_3. Cependant, l'action de réception pour le m_3 n'existe pas puisqu'il n'est pas couplé. Les ordres partiels induits par ces deux MSC sont donc les mêmes et ils sont donc égaux.

Nous pouvons lier les automates SR au système : tout mot d'un langage d'un automate $\mathtt{SR}(\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin})$ correspond bien à une séquence d'actions jouables dans le système, si l'on

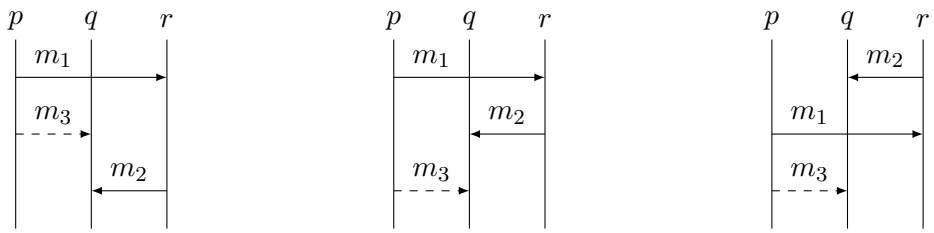


FIGURE 5.6 – MSC des échanges possibles

ignore le contenu des canaux. Et inversement, si une séquence d'actions composée d'envois puis de réceptions, permet d'aller de l'état $\overrightarrow{in}$ à l'état $\overrightarrow{fin}$, alors, on retrouvera le mot correspondant dans le langage de l'automate $\mathtt{SR}(\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin})$. Rappelons que $\overrightarrow{\ell} \overset{\mu}{\rightsquigarrow} \overrightarrow{\ell}'$ est valable s'il existe une linéarisation e de μ telle que $\overrightarrow{\ell} \overset{e}{\Rightarrow} \overrightarrow{\ell}'$.

Lemme 5.2.1. *Soit $w \in \Sigma^*$, $w \in \mathcal{L}(\mathtt{SR}(\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin}))$ pour un certain état $\overrightarrow{mid}$ si et seulement si $\overrightarrow{in} \overset{msc(w)}{\rightsquigarrow} \overrightarrow{fin}$.*

Démonstration.

Nous pouvons observer que, par construction de $\mathtt{SR}(\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin})$, $\ell^0_{\mathtt{SR}} \overset{w}{\Rightarrow} (\overrightarrow{\ell}, \overrightarrow{\ell}')$ si et seulement si $\overrightarrow{in} \overset{\sigma_1(w)}{\Longrightarrow} \overrightarrow{\ell}$ et $\overrightarrow{mid} \overset{\sigma_2(w)}{\Longrightarrow} \overrightarrow{\ell}'$ (ceci peut facilement être montré par récurrence sur la longueur de w). En particulier, w est accepté si et seulement si $\overrightarrow{in} \overset{\sigma_1(w)}{\Longrightarrow} \overrightarrow{mid}$ et $\overrightarrow{mid} \overset{\sigma_2(w)}{\Longrightarrow} \overrightarrow{fin}$, ce qui implique $\overrightarrow{in} \overset{msc(w)}{\rightsquigarrow} \overrightarrow{fin}$.

□

5.2.2 Automates de séquences valides

Nous l'avons dit, les séquences obtenues par les automates SR ont la forme d'un échange, mais nous ne savons rien sur leur réalisabilité. La reconnaissance des séquences d'actions mb-réalisables est alors nécessaire, elle se fera grâce à un nouveau type d'automates. Du produit de ces deux types d'automates résultera les mots encodant les échanges mb-réalisables du système.

Pour ce nouveau type d'automates, nous procéderons comme nous l'avons fait dans le chapitre précédent. À défaut de pouvoir se souvenir de tous les messages de toutes les séquences précédentes, nous créons les ensembles $\mathcal{C}_{\mathtt{S},p}$ et $\mathcal{C}_{\mathtt{R},p}$ permettant de se souvenir des chemins SS présents dans le graphe de dépendances étendu à partir d'un message non couplé envoyé à p. Nous pouvons alors décider si la séquence observée est mb-réalisable compte tenu des séquences passées. Rappelons que la présence d'un cycle SS indique que le MSC observé n'est pas mb-réalisable, comme le prouve le Théorème 4.1.1.

On avait déjà $\mathcal{B}(p) = (\mathcal{C}_{\mathtt{S},p}, \mathcal{C}_{\mathtt{R},p})$ pour une séquence d'actions. Chose nouvelle, on pourra attribuer à un MSC μ une fonction $\mathcal{B}$. On notera alors $\mathcal{B}^\mu = (\mathcal{C}^\mu_{\mathtt{S},p}, \mathcal{C}^\mu_{\mathtt{R},p})_{p\in\mathbb{P}}$. On définit alors

$$\begin{aligned}\mathcal{C}^\mu_{\mathtt{S},p} =& \{\mathsf{proc}_{\mathtt{S}}(\mathbf{m}) \mid \mathbf{m}' \overset{\text{SS}}{\dashrightarrow} \mathbf{m} \ \&\ \mathbf{m}' \text{ est non couplé } \&\ \mathsf{proc}_{\mathtt{R}}(\mathbf{m}') = p \ \&\\ & \mathbf{m}, \mathbf{m}' \in V \text{ avec } \mathsf{GDE}(\mu) = (V, E)\}\\ \mathcal{C}^\mu_{\mathtt{R},p} =& \{\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) \mid \mathbf{m}' \overset{\text{SS}}{\dashrightarrow} \mathbf{m} \ \&\ \mathbf{m}' \text{ est non couplé } \&\ \mathsf{proc}_{\mathtt{R}}(\mathbf{m}') = p \ \&\ \mathbf{m} \text{ est couplé}\\ & \mathbf{m}, \mathbf{m}' \in V \text{ avec } \mathsf{GDE}(\mu) = (V, E)\}\end{aligned}$$

Ainsi, on sait que si $\mathsf{GDE}(\mu)$ ne contient pas de cycle SS si, pour tout $p \in \mathbb{P}$, $p \notin \mathcal{C}^\mu_{\mathtt{R},p}$, ce qui est une conséquence immédiate du Théorème 4.1.1. Enfin, comme dans le chapitre précédent, nous notons $\mathbb{B} = (2^{\mathbb{P}} \times 2^{\mathbb{P}})^{\mathbb{P}}$ l'ensemble des $\mathcal{B}$ et nous ajoutons $\mathbb{B}_{good} = \{(\mathcal{C}_{\mathtt{S},p}, \mathcal{C}_{\mathtt{R},p})_{p\in\mathbb{P}} \mid \forall p, p \notin \mathcal{C}_{\mathtt{R},p}\}$ les $\mathcal{B}$ résultant d'une séquence mb-réalisable.

On peut alors déduire du Lemme 4.2.1 et des renommages ci-dessus la propriété suivante.

Propriété 3. *Pour $w \in \Sigma^*$, $msc(w)$ est* mb*-réalisable si et seulement si $\mathcal{B}^{msc(w)} \in \mathbb{B}_{good}$.*

Notons que $\mathbb{B}_{good}$ est un ensemble fini, nous sommes alors capable de construire un automate $\mathtt{MR}(\mathcal{B}, \mathcal{B}')$ avec $\mathcal{B}, \mathcal{B}' \in \mathbb{B}$. Intuitivement, $\mathcal{B}$ est l'état des canaux qui résume le graphe de dépendances dérivé des échanges précédents et $\mathcal{B}'$ celui obtenu quand le nouvel échange a été ajouté.

Définition 5.2.2 (Automate de séquences mb-réalisables)**.** L'automate $\mathtt{MR}(\mathcal{B}, \mathcal{B}')$ est défini comme suit :

- $\mathbb{B}$ est l'ensemble des états ;
- $\mathcal{B} = (\mathcal{C}_{\mathtt{S},p}, \mathcal{C}_{\mathtt{R},p})_{p\in\mathbb{P}}$ est l'état initial ;
- $\{\mathcal{B}'\}$ est l'ensemble des états finaux ;
- la relation de transition $(\overset{a}{\rightarrow})_{a\in\Sigma}$ est définie ainsi :
  - il existe $(\mathcal{C}^{(i)}_{\mathtt{S},p}, \mathcal{C}^{(i)}_{\mathtt{R},p})_{p\in\mathbb{P}} \xrightarrow{!?m^{p\to q}} (\mathcal{C}^{(i+1)}_{\mathtt{S},p}, \mathcal{C}^{(i+1)}_{\mathtt{R},p})_{p\in\mathbb{P}}$ si, pour tout $r \in \mathbb{P}$:
    - $\mathcal{C}^{(i+1)}_{\mathtt{S},r} = \mathcal{C}^{(i)}_{\mathtt{S},r} \cup \begin{cases} \{p\} \cup \mathcal{C}^{(i)}_{\mathtt{S},q} & \text{si } A \text{ ou } B \\ \emptyset & \text{sinon} \end{cases}$
    - $\mathcal{C}^{(i+1)}_{\mathtt{R},r} = \mathcal{C}^{(i)}_{\mathtt{R},r} \cup \begin{cases} \{q\} \cup \mathcal{C}^{(i)}_{\mathtt{R},q} & \text{si } A \text{ ou } B \\ \emptyset & \text{sinon} \end{cases}$
  - il existe $(\mathcal{C}^{(i)}_{\mathtt{S},p}, \mathcal{C}^{(i)}_{\mathtt{R},p})_{p\in\mathbb{P}} \xrightarrow{!m^{p\to q}} (\mathcal{C}^{(i+1)}_{\mathtt{S},p}, \mathcal{C}^{(i+1)}_{\mathtt{R},p})_{p\in\mathbb{P}}$ si, pour tout $r \in \mathbb{P}$:
    - $\mathcal{C}^{(i+1)}_{\mathtt{S},r} = \mathcal{C}^{(i)}_{\mathtt{S},r} \cup \begin{cases} \{p\} & \text{si } q = r \text{ ou } B \\ \emptyset & \text{sinon} \end{cases}$
    - $\mathcal{C}^{(i+1)}_{\mathtt{R},r} = \mathcal{C}^{(i)}_{\mathtt{R},r}$

  pour $A = p \in \mathcal{C}_{\mathtt{R},r} \cup \mathcal{C}^{(i)}_{\mathtt{S},r}$ et pour $B = q \in \mathcal{C}^{(i)}_{\mathtt{R},r}$.

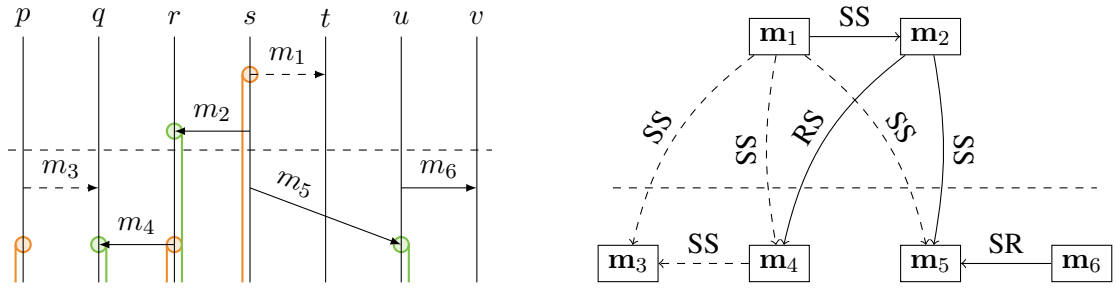
Comme à notre habitude, le langage reconnu par l'automate $\mathtt{MR}(\mathcal{B}, \mathcal{B}')$ est noté $\mathcal{L}(\mathtt{MR}(\mathcal{B}, \mathcal{B}'))$. Comme on peut le constater, les ensembles abstrayant les contenus des canaux sont les mêmes mais la façon d'y ajouter des processus diffère légèrement. En effet, puisque le MSC n'est pas observé comme une séquence d'actions mais comme une séquence de messages couplés ou non, les réceptions sont donc observées avant qu'elles puissent avoir lieu. Les envois des messages qui suivront auront en réalité lieu avant. Ainsi, deux cas particuliers sont à distinguer. L'on peut tout d'abord se questionner sur la condition A et en particulier "$p \in \mathcal{C}_{\mathtt{R},r}$". En effet, on ne considère pas que les messages où $p \in \mathcal{C}^{(i)}_{\mathtt{R},r}$ uniquement aient un lien SS avec un message non couplé envoyé à r. Ainsi, nous différencions l'état de $\mathcal{B}$ au moment du message ($\mathcal{C}^{(i)}_{\mathtt{R},r}$) et celui au début du k-échange ($\mathcal{C}_{\mathtt{R},r}$), pour ne pas créer un lien qui n'aurait pas lieu d'être. On observera un exemple de ce cas avec le message m_6 de l'Exemple 5.2.3. Le deuxième cas particulier concerne l'ajout de l'ensemble $\mathcal{C}^{(i)}_{\mathtt{S},q}$ à l'ensemble $\mathcal{C}^{(i+1)}_{\mathtt{S},r}$. Si un message non couplé a été envoyé à q précédemment, et que m qui est couplé est envoyé à q, alors, m doit être envoyé avant le message non couplé. Ainsi, tous les messages accessibles avec un chemin SS depuis ce message non couplé seront également accessibles depuis m. Ce cas sera illustré avec les messages m_3 et m_4 de l'Exemple 5.2.3.

Exemple 5.2.3 – Considérons le MSC $\mu = \mu_1 \cdot msc(w)$ de la Figure 5.7.a avec

$$w = !m_3^{p\to q}!?m_4^{r\to q}!?m_5^{s\to u}!?m_6^{u\to v}$$

Supposons que l'on commence à regarder $msc(w)$ avec $\mathcal{B}$ tel que $\mathcal{C}_{\mathtt{S},t} = \{s\}$ et $\mathcal{C}_{\mathtt{R},t} = \{r\}$ dû à μ_1. Alors, la mise à jour des variables $\mathcal{C}_{\mathtt{S},t}$, $\mathcal{C}_{\mathtt{R},t}$ et $\mathcal{C}_{\mathtt{S},q}$, après avoir lu chaque lettre du mot w est décrite ci-dessous.

		$!m_3^{p\to q}$		$!?m_4^{r\to q}$		$!?m_5^{s\to u}$		$!?m_6^{u\to v}$	
$\mathcal{C}_{\mathtt{S},t}$	$\{s\}$	→	$\{s\}$	→	$\{p,r,s\}$	→	$\{p,r,s\}$	→	$\{p,r,s\}$
$\mathcal{C}_{\mathtt{R},t}$	$\{r\}$		$\{r\}$		$\{q,r\}$		$\{q,r,u\}$		$\{q,r,u\}$
$\mathcal{C}_{\mathtt{S},q}$	$\emptyset$		$\{p\}$		$\{p\}$		$\{p\}$		$\{p\}$



(a) **(b)**

FIGURE 5.7 – MSC μ avec l'ensemble $\mathcal{C}_{\mathtt{S},t}$ en orange et $\mathcal{C}_{\mathtt{R},t}$ en vert (a) et le graphe de dépendances étendu associé (b)

Décrivons en détail les modifications effectuées. Le lecteur pourra se référer au graphe de dépendances étendu de μ à la Figure 5.7.b permettant de visualiser les dépendances inapparentes dans le MSC.

- Le message m_3 est envoyé par p à q. Comme il n'est pas couplé, l'expéditeur est ajouté à l'ensemble $\mathcal{C}_{\mathtt{S},q}$ du destinataire, et alors $p \in \mathcal{C}^{(1)}_{\mathtt{S},q}$.

- Le message m_4 est envoyé par r à q. $r \in \mathcal{C}_{\mathtt{R},t}$, donc un message couplé reçu par r dans un k-échange précédent dépendait d'un message non couplé envoyé à t. Comme m_4 est couplé, on ajoute alors r à $\mathcal{C}^{(2)}_{\mathtt{S},t}$ et q à $\mathcal{C}^{(2)}_{\mathtt{R},t}$. Tout message envoyé par r après cette réception aura lui aussi été envoyé après ce message non couplé à t. Et tout message envoyé ou reçu par q à partir de cet instant aura été envoyé après m_4. On ajoute aussi l'ensemble $\mathcal{C}^{(1)}_{\mathtt{S},q} = \{p\}$ à $\mathcal{C}_{\mathtt{S},t}$, les messages dépendants de m_1 dépendront aussi alors de m_4, ce qui est alors le cas de m_3. On retrouve cette dépendance avec l'arc SS entre les sommets $\mathbf{m}_1$ et $\mathbf{m}_3$ dans le graphe de dépendances étendu. Finalement, $\mathcal{C}^{(2)}_{\mathtt{S},t} = \{p, r, s\}$ et $\mathcal{C}^{(2)}_{\mathtt{R},t} = \{q, r\}$.

- Le message m_5 est envoyé par le processus $s \in \mathcal{C}^{(2)}_{\mathtt{S},t}$. Ainsi on devrait ajouter à $\mathcal{C}_{\mathtt{S},t}$ l'ensemble $\mathcal{C}_{\mathtt{S},u} = \emptyset$ mais celui-ci est vide car aucun message non couplé n'a été envoyé à u. Le destinataire u est ajouté à l'ensemble $\mathcal{C}_{\mathtt{R},t}$. Ainsi, $\mathcal{C}^{(3)}_{\mathtt{R},t} = \{q, r, u\}$.

- Enfin, le message m_6 est envoyé par le processus $u \in \mathcal{C}^{(3)}_{\mathtt{R},t}$. Cependant, si l'expéditeur appartient à un ensemble $\mathcal{C}^{(i)}_{\mathtt{R},\cdot}$ mais pas à l'ensemble $\mathcal{C}_{\mathtt{R},\cdot}$, il n'est pas ajouté à l'ensemble $\mathcal{C}_{\mathtt{S},t}$. Ceci est dû au fait que m_6 peut très bien être envoyé avant m_5, il n'y a pas de lien causal entre leurs envois.

On prouve alors que l'automate $\mathtt{MR}(\mathcal{B}, \mathcal{B}')$ reconnaît en effet les mots w tels que si $msc(w)$ commence avec des canaux dans un état $\mathcal{B}$, il finira avec des canaux dans un état $\mathcal{B}'$.

Lemme 5.2.2. *Soient $\mathcal{B}, \mathcal{B}' \in \mathbb{B}$ et $w \in \Sigma^*$. Alors, $w \in \mathcal{L}(\mathtt{MR}(\mathcal{B}, \mathcal{B}'))$ si et seulement si pour tout MSC μ tel que $\mathcal{B} = \mathcal{B}^\mu$, $\mathcal{B}' = \mathcal{B}^{\mu \cdot msc(w)}$.*

Démonstration.

Prenons $w = a_0 \cdots a_n \in \Sigma^*$. Pour prouver ce lemme, il suffit de démontrer que si $\mathcal{B} = \mathcal{B}^\mu$ et

$$(\mathcal{C}^{(0)}_{\mathtt{S},p}, \mathcal{C}^{(0)}_{\mathtt{R},p})_{p\in\mathbb{P}} \xrightarrow{a_0} (\mathcal{C}^{(1)}_{\mathtt{S},p}, \mathcal{C}^{(1)}_{\mathtt{R},p})_{p\in\mathbb{P}} \xrightarrow{a_1} \cdots \xrightarrow{a_n} (\mathcal{C}^{(n+1)}_{\mathtt{S},p}, \mathcal{C}^{(n+1)}_{\mathtt{R},p})_{p\in\mathbb{P}} = \mathcal{B}'$$

alors $\mathcal{B}' = \mathcal{B}^{\mu \cdot msc(w)}$.

Il s'agit d'une preuve par récurrence sur la longueur de w, où l'hypothèse de récurrence est la suivante

$$(\mathcal{C}^{(n)}_{\mathtt{S},p}, \mathcal{C}^{(n)}_{\mathtt{R},p})_{p\in\mathbb{P}} = \mathcal{B}^{\mu \cdot msc(a_0 \cdots a_{n-1})}$$

Commençons par démontrer que, $\forall r \in \mathbb{P}, \mathcal{C}^{(n+1)}_{\mathtt{S},r} = \mathcal{C}^{\mu \cdot msc(w)}_{\mathtt{S},r}$. Supposons que $p \in \mathcal{C}^{(n+1)}_{\mathtt{S},r}$. Si $p \in \mathcal{C}^{(n+1)}_{\mathtt{S},r}$ alors on peut immédiatement conclure que $p \in \mathcal{C}^{\mu \cdot msc(w)}_{\mathtt{S},r}$ puisque $\mathcal{C}^{(0)}_{\mathtt{S},r} = \mathcal{C}^{\mu}_{\mathtt{S},r} \subseteq \mathcal{C}^{(n+1)}_{\mathtt{S},r}$ et que la fonction $\mathcal{B}^\cdot$ est croissante et monotone. Dans le cas contraire, si $p \in \mathcal{C}^{(0)}_{\mathtt{S},r}$ alors les cas suivants peuvent arriver (sans perte de généralité, on suppose que p a été ajouté lors de la lecture du dernier symbole de w) :

1. $a_n = !?m^{p\to q}$ et $p \in \mathcal{C}^{(0)}_{\mathtt{R},r} = \mathcal{C}^{\mu}_{\mathtt{R},r}$:

   Alors il existe un message m'' dans μ tel que $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}'') = p$ et $\mathbf{m}' \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}''$ avec $\mathbf{m}'$ non couplé. Alors, il est facile de voir que $\mathbf{m}'' \overset{\mathrm{SS}}{\dashrightarrow} \mathbf{m}$ et donc $p \in \mathcal{C}^{\mu \cdot msc(w)}_{\mathtt{S},r}$;

2. $a_n = !?m^{p\to q}$ et $q \in \mathcal{C}^{(n)}_{\mathtt{R},r} = \mathcal{C}^{\mu\cdot msc(a_0\cdots a_{n-1})}_{\mathtt{R},r}$:
   Alors il existe un message m'' dans $\mu \cdot msc(a_0 \cdots an-1)$ tel que $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}'') = q$ et $\mathbf{m}' \overset{\text{SS}}{\dashrightarrow} \mathbf{m}''$ avec $\mathbf{m}'$ non couplé. Alors, il est facile de voir que $\mathbf{m}'' \overset{\text{SS}}{\dashrightarrow} \mathbf{m}$ et donc $p \in \mathcal{C}^{\mu\cdot msc(w)}_{\mathtt{S},r}$;
3. $a_n = !?m^{p'\to q}$ et $p' \in \mathcal{C}_{\mathtt{R},r} \cup \mathcal{C}^{(n)}_{\mathtt{S},r}$ et $p \in \mathcal{C}^{(n)}_{\mathtt{S},q} = \mathcal{C}^{\mu\cdot msc(a_0\cdots a_{n-1})}_{\mathtt{S},q}$.
   Alors il existe un message m'' dans $\mu \cdot msc(a_0 \cdots a_{n-1})$, tel que $\mathsf{proc}_{\mathtt{S}}(\mathbf{m}'') = p$ et $\mathbf{m}' \overset{\text{SS}}{\dashrightarrow} \mathbf{m}''$ avec $\mathbf{m}'$ non couplé et $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}') = q$. Alors, il est facile de voir que $\mathbf{m} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}''$ et, puisque $p' \in \mathcal{C}_{\mathtt{R},r} \cup \mathcal{C}^{(n)}_{\mathtt{S},r}$ et avec une analyse similaire à ci-dessus, on a $\mathbf{m}''' \overset{\text{SS}}{\dashrightarrow} \mathbf{m}'''$ avec $\mathbf{m}'''$ non couplé, et l'on peut donc conclure que $p \in \mathcal{C}^{\mu\cdots msc(w)}_{\mathtt{S},r}$.
4. $a_n = !m^{p\to q}$ et $p \in \mathcal{C}_{\mathtt{R},r}$
   Cas analogue au $2^{\text{ème}}$ cas ci-dessus.
5. $a_n = !m^{p\to r}$
   Dans ce cas, on peut immédiatement conclure que $p \in \mathcal{C}^{\mu\cdot msc(w)}_{\mathtt{S},r}$ puisque m est non couplé et que $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = r$.

Maintenant, supposons que $p \in \mathcal{C}^{\mu\cdot msc(w)}_{\mathtt{S},r}$ (on peut supposer que $p \notin \mathcal{C}^{\mu\cdot msc(a_0\cdots a_{n-1})}_{\mathtt{S},r}$ sans perte de généralité). Alors,

- soit $a_n = !m^{p\to q}$, et donc on voit immédiatement que $p \in \mathcal{C}^{(n+1)}_{\mathtt{S},r}$,
- soit $a_n = !?m^{p\to q}$ et $\mathbf{m}' \overset{\text{SS}}{\dashrightarrow} \mathbf{m}$ pour un certain $\mathbf{m}'$ non couplé et $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}') = r$.

$p \notin \mathcal{C}^{\mu\cdot msc(a_0\cdots a_{n-1})}_{\mathtt{S},r}$ implique que

- soit $q \in \mathcal{C}^{\mu\cdot msc(a_0\cdots a_{n-1})}_{\mathtt{R},r}$
- soit $p \in \mathcal{C}^{\mu}_{\mathtt{R},r}$ (notez que, puisque w est un échange, $p \notin \mathcal{C}^{\mu\cdot msc(a_0\cdots a_{n-1})}_{\mathtt{R},r} \setminus \mathcal{C}^{\mu}_{\mathtt{R},r}$).

Dans les deux cas, on peut conclure que $p \in \mathcal{C}^{(n+1)}_{\mathtt{S},r}$.

Montrons à présent que $\forall r \in \mathbb{P}, \mathcal{C}'_{\mathtt{R},r} = \mathcal{C}^{\mu\cdot msc(w)}_{\mathtt{R},r}$. Supposons que $p \in \mathcal{C}^{(n+1)}_{\mathtt{R},r}$ (on peut supposer que $p \notin \mathcal{C}^{(n+1)}_{\mathtt{R},r}$ sans perte de généralité). Ceci implique que :

- $a_n = !?m^{q\to p}$ avec $q \in \mathcal{C}_{\mathtt{R},r} \cup \mathcal{C}^{(n+1)}_{\mathtt{S},r} = \mathcal{C}^{\mu\cdot msc(w)}_{\mathtt{S},r}$.
  Alors il existe un message $\mathbf{m}'' \in \mu \cdot msc(w)$ tel que $\mathsf{proc}_{\mathtt{S}}(\mathbf{m}'') = q$ et $\mathbf{m}' \overset{\text{SS}}{\dashrightarrow} \mathbf{m}''$ avec $\mathbf{m}'$ non couplé et $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}') = r$. Alors il est facile de voir que $\mathbf{m}'' \overset{\text{SS}}{\dashrightarrow} \mathbf{m}$ et nous pouvons conclure que $p \in \mathcal{C}^{\mu\cdot msc(w)}_{\mathtt{R},r}$.
- $a_n = !?m^{q\to p'}$ avec $q \in \mathcal{C}_{\mathtt{R},r} \cup \mathcal{C}^{(n+1)}_{\mathtt{S},r} = \mathcal{C}^{\mu\cdot msc(w)}_{\mathtt{S},r}$ et $p \in \mathcal{C}^{(n)}_{\mathtt{R},p'} = \mathcal{C}^{\mu\cdot msc(a_0\cdots a_{n-1})}_{\mathtt{R},p'}$
  Alors il existe un message $\mathbf{m}''$ dans $\mu \cdot msc(w)$ tel que $\mathsf{proc}_{\mathtt{S}}(\mathbf{m}'') = q$ et $\mathbf{m}' \overset{\text{SS}}{\longrightarrow} \mathbf{m}''$ avec $\mathbf{m}'$ non couplé et $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}') = r$. De la même façon, il existe $\mathbf{m}'''$ dans $\mu \cdot msc(a_0 \cdots a_{n-1})$ tel que $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}''') = p$ et $\mathbf{m}^{iv} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}'''$ avec $\mathbf{m}^{iv}$ non couplé et $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}^{iv}) = p'$. Maintenant, quand on ajoute $\mathbf{m}$ au graphe de conflits, on obtient $\mathbf{m} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}^{iv}$. On peut donc conclure $p \in \mathcal{C}^{\mu\cdot msc(w)}_{\mathtt{R},r}$.

Maintenant, supposons que $p \in \mathcal{C}_{\mathtt{R},r}^{\mu \cdot msc(w)}$ (on peut supposer que $p \notin \mathcal{C}_{\mathtt{R},r}^{\mu \cdot msc(a_0 \cdots a_{n-1})}$ sans perte de généralité). On sait que $\mathcal{C}_{\mathtt{R},r}^{\mu \cdot msc(a_0 \cdots a_{n-1})} = \mathcal{C}_{\mathtt{R},r}^{(n)}$. Posons $a_n = !?m^{q \to p}$. Quatre cas sont alors possibles :

- $q \in \mathcal{C}_{\mathtt{S},r}^{\mu \cdot msc(a_0 \cdots a_{n-1})}$,
- ou $q \in \mathcal{C}_{\mathtt{R},r}^{\mu \cdot msc(a_0 \cdots a_{n-1})}$,
- ou $p \in \mathcal{C}_{\mathtt{S},r}^{\mu \cdot msc(a_0 \cdots a_{n-1})}$,
- ou $a_n = !?m^{q' \to p'}$ et $p \in \mathcal{C}_{\mathtt{R},p'}^{\mu \cdot msc(a_0 \cdots a_{n-1})}$ et $q' \in \mathcal{C}_{\mathtt{S},r}^{\mu \cdot msc(a_0 \cdots a_{n-1})}$.

Pour tous ces cas, par hypothèse de récurrence et par Définition 5.2.2, on peut conclure que $p \in \mathcal{C}_{\mathtt{R},r}^{(n+1)}$.

□

On peut alors aussi déduire la proposition suivante. Comme dans le chapitre précédent, on utilisera la notation $\mathcal{B}_\emptyset$ pour l'abstraction des canaux dans l'état initial $(\emptyset, \emptyset)_{p \in \mathbb{P}}$.

Proposition 5.2.3. *Pour tout MSC μ, $\mu \in \textsc{Msc}(\mathcal{S})$ si et seulement si il existe $\vec{\ell}$ et $\mathcal{B} \in \mathbb{B}_{good}$ tels que $(\vec{\ell_0}, \mathcal{B}_\emptyset) \overset{\mu}{\rightsquigarrow} (\vec{\ell}, \mathcal{B})$.*

Remarque 5.2.1 – On constate que l'automate MR ne gère pas les cas illustrés dans la Figure 5.8. En effet, ces cas là sont déjà gérés par l'automate SR qui, en manipulant les deux actions d'un message en même temps, si elles existent, s'assurent qu'une telle situation ne peut jamais arriver. En effet, aucun mot dans Σ^* ne peut représenter ce genre de situation et l'automate MR n'a alors qu'à gérer les cas liés aux messages non couplés.

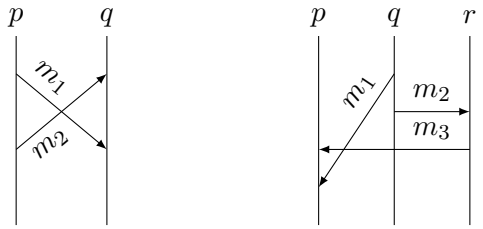


FIGURE 5.8 – Des MSC pas mb-réalisables

5.2.3 Langage d'échanges accessibles

Avant de s'intéresser aux échanges premiers, il nous reste à combiner les automates construits dans les deux dernières sections : celui qui reconnaît les mots encodant les échanges, et celui qui reconnaît les mots encodant des séquences valides. En effet, le langage $\mathcal{L}(\mathtt{SR}(\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin}))$ contient des échanges qui ne sont pas tous mb-réalisables. Les intersections avec les automates $\mathtt{MR}(\mathcal{B}, \mathcal{B}')$, avec $\mathcal{B}$ et $\mathcal{B}' \in \mathbb{B}_{good}$ permettrons de restreindre l'ensemble des échanges et d'obtenir ceux que l'on peut trouver dans le système et qui sont mb-réalisables. Ainsi, nous noterons

$\mathcal{L}_f(\overrightarrow{in}, \overrightarrow{fin}, \mathcal{B}, \mathcal{B}')$ un langage d'échanges mb-réalisables partant de l'état $\overrightarrow{in}$, passant par un état $\overrightarrow{mid}$ et finissant dans l'état $\overrightarrow{fin}$ et nous les définissons comme suit :

$$\mathcal{L}_f(\overrightarrow{in}, \overrightarrow{fin}, \mathcal{B}, \mathcal{B}') \stackrel{\text{def}}{=} \bigcup_{\overrightarrow{mid} \in L_{\mathcal{S}}} \mathcal{L}(\mathtt{SR}(\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin})) \cap \mathcal{L}(\mathtt{MR}(\mathcal{B}, \mathcal{B}'))$$

Intuitivement, il s'agit du langage contenant les (encodages des) échanges entre les états de contrôle $\overrightarrow{in}$ et $\overrightarrow{fin}$, commençant avec une certaine fonction $\mathcal{B}$ et finissant avec une autre fonction $\mathcal{B}'$. De plus, comme $\mathcal{B}$ et $\mathcal{B}'$ sont contraints à appartenir à $\mathbb{B}_{good}$, ces échanges sont mb-réalisables.

On peut à présent combiner les échanges valides qui peuvent être exécutés les uns après les autres à partir de l'état initial du système $\overrightarrow{\ell_0}$. Tel est l'objectif de la définition de l'ensemble $\mathcal{A}$ des *langages accessibles*.

Définition 5.2.3 (Langages accessibles)**.** Pour un système $\mathcal{S} = ((L_p, \delta_p, \ell_p^0)_{p \in \mathbb{P}}, \mathsf{com})$ donné, l'ensemble d'échanges accessibles $\mathcal{A}$ est le plus petit ensemble de langages $\mathcal{L}_f(\overrightarrow{in}, \overrightarrow{fin}, \mathcal{B}_i, \mathcal{B}_f)$ défini comme suit :

1. pour tout $\overrightarrow{\ell} \in L_{\mathcal{S}}$ et $\mathcal{B} \in \mathbb{B}_{good}$, le langage $\mathcal{L}_f(\overrightarrow{\ell_0}, \overrightarrow{\ell}, \mathcal{B}_\emptyset, \mathcal{B}) \in \mathcal{A}$;
2. pour tout $\overrightarrow{\ell_1}, \overrightarrow{\ell_2}, \overrightarrow{\ell_3} \in L_{\mathcal{S}}$ et $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3 \in \mathbb{B}_{good}$, si le langage $\mathcal{L}_f(\overrightarrow{\ell_1}, \overrightarrow{\ell_2}, \mathcal{B}_1, \mathcal{B}_2) \in \mathcal{A}$ et $\mathcal{L}_f(\overrightarrow{\ell_1}, \overrightarrow{\ell_2}, \mathcal{B}_1, \mathcal{B}_2) \neq \emptyset$ alors $\mathcal{L}_f(\overrightarrow{\ell_2}, \overrightarrow{\ell_3}, \mathcal{B}_2, \mathcal{B}_3) \in \mathcal{A}$.

Ainsi, l'union $\bigcup \mathcal{A}$ de tous les langages accessibles est égal au langage

$$\mathcal{L}_a = \{w \in \Sigma^* \mid msc(w) \text{ est accessible } \}$$

Par conséquent, l'on obtient le résultat suivant.

Théorème 5.2.4. *$\mathcal{L}_a$ est un langage régulier et est accepté par un automate fini.*

Démonstration.

$\Rightarrow w \in \bigcup \mathcal{A}$ donc il existe une séquence de mots $w_1 \cdots w_n \in \Sigma^*$ telle que $\forall 1 \leq j \leq n, w_j \in \mathcal{L}_f(\overrightarrow{\ell}_{j-1}, \overrightarrow{\ell}_j, \mathcal{B}_{j-1}, \mathcal{B}_j) \neq \emptyset$ avec $\mathcal{B}_j \in \mathbb{B}_{good}$, et il existe aussi $\overrightarrow{fin} \in L_{\mathcal{S}}, \mathcal{B}_f \in \mathbb{B}_{good}$ tels que $w \in \mathcal{L}_f(\overrightarrow{\ell}_n, \overrightarrow{fin}, \mathcal{B}_n, \mathcal{B}_f)$.

Donc, $\forall j, w_j \in \mathcal{L}(\mathtt{SR}(\overrightarrow{\ell_{j-1}}, \overrightarrow{\ell_j}', \overrightarrow{\ell_j}))$ pour $\overrightarrow{\ell_j}' \in L_{\mathcal{S}}$ et $w \in \mathcal{L}(\mathtt{SR}(\overrightarrow{\ell_n}, \overrightarrow{\ell_n}', \overrightarrow{fin}))$. D'après le Lemme 5.2.1, $\overrightarrow{\ell_{j-1}} \overset{msc(w_j)}{\rightsquigarrow} \overrightarrow{\ell_j}$ et $\overrightarrow{\ell_n} \overset{msc(w)}{\rightsquigarrow} \overrightarrow{fin}$.

De plus, $\forall j$, $w_j \in \mathcal{L}(\mathtt{MR}(\mathcal{B}_{j-1}, \mathcal{B}_j))$ et $\mathcal{B}_j \in \mathbb{B}_{good}$. Donc, tout $msc(w_j)$ est mb-réalisable et même $msc(w_1) \cdots msc(w_n) \cdot msc(w)$ est mb-réalisable. Finalement, d'après la Proposition 5.2.3, $msc(w_1) \cdots msc(w_n) \cdot msc(w) \in \textsc{Msc}(\mathcal{S})$ et donc, $msc(w)$ est accessible.

$\Leftarrow msc(w)$ est accessible donc il existe une séquence de MSC $\mu_1 \cdots \mu_n$ telle que $\mu_1 \cdots \mu_n \cdot msc(w) \in \textsc{Msc}(\mathcal{S})$.

- ✢ Supposons que $\mu_1 \cdots \mu_n = \varepsilon$, alors, $\overrightarrow{\ell_0} \overset{msc(w)}{\rightsquigarrow} \overrightarrow{fin}$ pour un certain $\overrightarrow{fin}$. Ainsi, $w \in \mathcal{L}(\mathtt{SR}(\overrightarrow{\ell_0}, \overrightarrow{\ell}, \overrightarrow{fin}))$ pour un certain $\overrightarrow{\ell} \in L_{\mathcal{S}}$. Comme $msc(w) \in \textsc{Msc}(\mathcal{S})$, $msc(w)$ est mb-réalisable donc il existe $\mathcal{B}$ tel que $w \in \mathcal{L}(\mathtt{MR}(\mathcal{B}_0, \mathcal{B}'))$ avec $\mathcal{B}' \in \mathbb{B}_{good}$. Finalement, on a donc que $w \in \mathcal{L}_f(\overrightarrow{\ell_0}, \overrightarrow{fin}, \mathcal{B}_0, \mathcal{B})$ et donc $w \in \bigcup \mathcal{A}$.

✢ Maintenant, supposons que $\mu_1 \cdots \mu_n \neq \varepsilon$. Alors, il existe une séquence $w_1, \cdots, w_n \in \Sigma^*$ telle que $msc(w_i) = \mu_i$, $1 \leq i \leq n$, et on peut supposer que $w_1, \cdots, w_n \in \mathcal{L}_a$. D'après le Lemme 5.2.2, il y a $\mathcal{B} = \mathcal{B}^{\mu_1 \cdots \mu_n}$, $\mathcal{B}' = \mathcal{B}^{\mu_1 \cdots \mu_n \cdot msc(w)} \in \mathbb{B}_{good}$ tel que $w \in \mathcal{L}(\mathtt{MR}(\mathcal{B}, \mathcal{B}'))$. De plus, il existe $\overrightarrow{in}, \overrightarrow{fin}$ tels que $\overrightarrow{in} \overset{msc(w)}{\rightsquigarrow} \overrightarrow{fin}$ alors $w \in \mathcal{L}(\mathtt{SR}(\overrightarrow{in}, \overrightarrow{mid}, \overrightarrow{fin}))$ pour un certain $\overrightarrow{mid}$. Enfin, on a bien $w \in \mathcal{L}_f(\overrightarrow{in}, \overrightarrow{fin}, \mathcal{B}, \mathcal{B}')$ et donc $w \in \mathcal{L}_a$.

□

5.3 Les échanges premiers

Nous avons donc construit les langages contenant les échanges accessibles du système. La prochaine étape consiste à réduire ce langage aux échanges accessibles et premiers, qui ne peuvent pas être divisés en échanges plus petits.

L'objectif final est, rappelons-le, de trouver la borne k, telle que toutes les exécutions peuvent être divisées en k-échanges. Ainsi, si un échange est de taille $K > k$ mais peut être divisé en deux k-échanges par exemple, ce premier échange ne nous intéresse pas : seul le plus petit découpage nous sera utile pour trouver k.

Nous pouvons reformuler la notion d'échange premier en termes de graphe de dépendances. Comme nous l'avons vu dans le chapitre précédent, les messages appartenant à la même composante fortement connexe doivent appartenir au même échange. Ainsi, si tous les messages sont accessibles les uns depuis les autres, ils appartiennent à la même composante et forment un et un seul échange, et ce même dans le plus petit découpage. On dira d'un graphe de dépendances $\mathsf{GD}(\mu)$ associé au MSC μ qu'il est *fortement connecté* si, pour tout $\mathbf{m}, \mathbf{m}' \in V$ il existe $\mathbf{m} \rightarrow^* \mathbf{m}'$, où $\rightarrow^*$ est la fermeture réflexive et transitive de $\rightarrow = \bigcup_{\mathrm{X,Y} \in \{\mathrm{S,R}\}} \xrightarrow{\mathrm{XY}}$.

Lemme 5.3.1. *Un échange μ est premier si et seulement si $\mathsf{GD}(\mu)$ est fortement connecté.*

Démonstration.

Soit $\mu = \mu_1 \cdots \mu_n$ un MSC formé d'une séquence d'échanges. Soient e, e' deux évènements de μ, et fixons $i, i' \in \{1, \cdots, n\}$ tels que e apparaît dans μ_i et e' dans $\mu_{i'}$. S'il existe un arc $e \xrightarrow{\mathrm{XY}} e'$ dans le graphe de dépendances de μ, alors $i \leq i'$. Ainsi, si e et e' sont dans la même composante fortement connexe, alors $i = i'$ et, si le graphe de dépendances de μ est fortement connecté, alors $n = 1$ et μ est un échange premier.

□

La prochaine étape consiste donc en la construction de l'automate capable de reconnaître le langage $\mathcal{L}_p = \{w \in \Sigma^* \mid msc(w) \text{ est premier}\}$. Nous utiliserons la caractérisation graphique établie avec le Lemme 5.3.1.

Comme précédemment, cet automate s'assurera uniquement du fait d'être premier et non pas du fait d'être un échange ou d'être mb-réalisable. Ainsi, cet automate travaillera sur des graphes de dépendances, pour lesquels nous pourrons déterminer s'ils représentent échange premier ou non. Comme il existe une infinité de graphes GD, et que notre but est d'obtenir un automate fini, nous utilisons des abstractions de ces graphes. Définissons cette abstraction.

Dans un premier temps, commençons par définir des transformations de graphe. Les graphes que nous allons manipuler sont des graphes orientés étiquetés avec deux ensembles de processus

sur chaque sommet. Nous appellerons de tels objets des *P-graphes*. Formellement, un P-graphe est un uplet $(V, E, \lambda_{\mathtt{S}}, \lambda_{\mathtt{R}})$ avec $E \subseteq V \times V$ et $\lambda_{\mathtt{X}} : V \to 2^{\mathbb{P}}$ pour $\mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}$. Le P-graphe $\mathsf{PG}(\mu)$ associé au graphe de dépendances $\mathsf{GD}(\mu) = (V, \{\xrightarrow{\mathrm{XY}}\}_{\mathrm{X,Y} \in \{\mathrm{S,R}\}})$ est décrit par l'uplet $(V, E, \lambda_{\mathtt{S}}, \lambda_{\mathtt{R}})$ où

1. $(\mathbf{m}, \mathbf{m}') \in E$ si $\mathbf{m} \xrightarrow{\mathrm{XY}} \mathbf{m}'$ pour certains X,Y ;
2. $\lambda_{\mathtt{S}}(\mathbf{m}) = \{\mathsf{proc}_{\mathtt{S}}(\mathbf{m})\}$;
3. si $\mathbf{m}$ est couplé, alors $\lambda_{\mathtt{R}}(\mathbf{m}) = \{\mathsf{proc}_{\mathtt{R}}(\mathbf{m})\}$ et si $\mathbf{m}$ n'est pas couplé, alors $\lambda_{\mathtt{R}}(\mathbf{m}) = \emptyset$.

La première transformation consiste à fusionner les sommets qui appartiennent à la même composante fortement connexe. Formellement, soit $G = (V, E, \lambda_{\mathtt{S}}, \lambda_{\mathtt{R}})$ un P-graphe alors $\mathsf{merge}(G) = (V', E', \lambda_{\mathtt{S}}, \lambda_{\mathtt{R}})$ est défini par

1. V' est l'ensemble des composantes fortement connexes maximales de G,
2. pour deux composantes fortement connexes distinctes U, U', $(U, U') \in E'$ s'il existe $\mathbf{m} \in U$ et $\mathbf{m}' \in U'$ tels que $(\mathbf{m}, \mathbf{m}') \in E^+$ (la fermeture transitive de E),
3. pour $\mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}$, $\lambda_{\mathtt{X}} = \bigcup_{\mathbf{m} \in U} \lambda_{\mathtt{X}}(\mathbf{m})$.

La deuxième transformation consiste à effacer les processus *redondants* dans les fonctions $\lambda_{\mathtt{S}}$ et $\lambda_{\mathtt{R}}$. Fixons $\mathbf{m} \in V$, $\mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}$ et $p \in \lambda_{\mathtt{X}}(\mathbf{m})$. On dit que p est $\mathtt{X}$-redondant dans $\mathbf{m}$ s'il existe $\mathbf{m}_1, \mathbf{m}_2$ tels que

1. $(\mathbf{m}_1, \mathbf{m}) \in E^+$ et $(\mathbf{m}, \mathbf{m}_2) \in E^+$ et
2. $p \in \lambda_{\mathtt{X}}(\mathbf{m}_1) \cap \lambda_{\mathtt{X}}(\mathbf{m}_2)$.

Intuitivement, p est redondant dans $\mathbf{m}$ s'il apparaît dans l'ensemble d'un ancêtre et dans celle d'un descendant de $\mathbf{m}$. On définit alors le P-graphe $\mathsf{erase}(G)$ comme $(V, E, \lambda'_{\mathtt{S}}, \lambda'_{\mathtt{R}})$ où pour tout $\mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}$ et pour tout $\mathbf{m} \in V$, $\lambda'_{\mathtt{X}}(\mathbf{m})$ est l'ensemble des processus $p \in \lambda_{\mathtt{X}}(\mathbf{m})$ tels que p n'est pas X-redondant dans $\mathbf{m}$. La dernière transformation consiste à supprimer les sommets qui auraient leurs ensembles vides. Formellement, pour $G = (V, E, \lambda_{\mathtt{S}}, \lambda_{\mathtt{R}})$, le P-graphe $\mathsf{sweep}(G)$ est $(V', E', \lambda_{\mathtt{S}}, \lambda_{\mathtt{R}})$ où

- $V' = \{\mathbf{m} \in V \mid \lambda_{\mathtt{S}}(\mathbf{m}) \cup \lambda_{\mathtt{R}}(\mathbf{m}) \neq \emptyset\}$ et
- $E' = E \cap V' \times V'$.

L'abstraction $\alpha(G)$ d'un P-graphe G est obtenu avec $\mathsf{sweep}(\mathsf{erase}(\mathsf{merge}(G)))$. Détaillons une telle construction dans l'exemple suivant.

Exemple 5.3.1 – Dans la Figure 5.9, l'on peut voir un MSC μ en (a) et son graphe de dépendances en (b).

Voyons les étapes pour obtenir l'abstraction de ce graphe de dépendances.

- Le graphe est tout d'abord transformé en P-graphe $\mathsf{PG}(\mu)$ que l'on retrouve en Figure 5.10. Celui-ci est similaire au graphe de dépendances, on ignore les étiquettes sur les arcs et l'on ajoute les ensembles $\lambda_{\mathtt{S}}$ et $\lambda_{\mathtt{R}}$ pour chaque processus. Ces ensembles sont représentés dans un tableau associé au P-graphe pour faciliter la lecture.

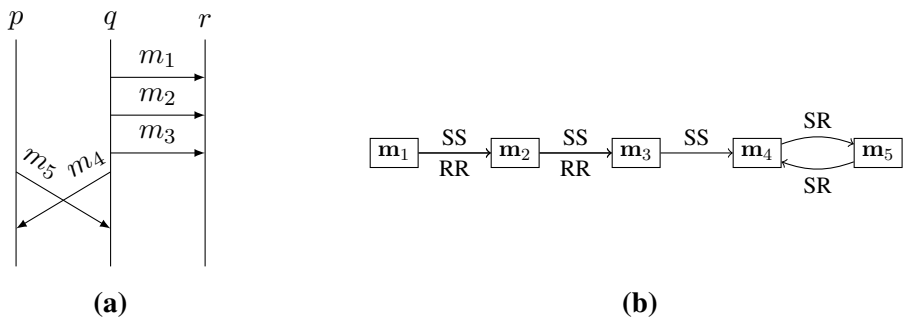


FIGURE 5.9 – MSC μ (a) et son graphe de dépendances (b)

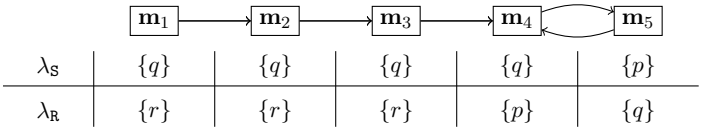


FIGURE 5.10 – $\mathsf{PG}(\mu)$

✢ La première étape, la fonction $\mathsf{merge}(\cdot)$, consiste à regrouper les sommets appartenant à la même composante fortement connexe, ainsi, on obtient un sommet contenant $\mathbf{m}_4$ et $\mathbf{m}_5$. Les ensembles λ_S et λ_R sont eux aussi fusionnés. On obtient alors $\mathsf{merge}(\mathsf{PG}(\mu))$ de la Figure 5.11.

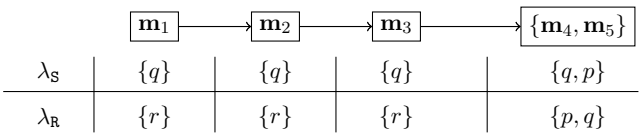


FIGURE 5.11 – $\mathsf{merge}(\mathsf{PG}(\mu))$

✢ La deuxième étape, la fonction $\mathsf{erase}(\cdot)$, supprime les processus redondants des ensembles λ_S et λ_R. Ainsi, on constate $q \in \lambda_S(\mathbf{m}_2)$ mais également que $q \in \lambda_S(\mathbf{m}_1)$ avec $\mathbf{m}_1$ un ancêtre de $\mathbf{m}_2$ (car $\mathbf{m}_1 \rightarrow \mathbf{m}_2$), et $q \in \lambda_S(\mathbf{m}_3)$ avec $\mathbf{m}_3$ un descendant de $\mathbf{m}_2$ (car $\mathbf{m}_2 \rightarrow \mathbf{m}_3$). Ainsi, q est dit S-redondant, et peut être supprimé. On peut faire le même raisonnement pour $q \in \lambda_S(\mathbf{m}_3)$ et $r \in \lambda_R(\mathbf{m}_2)$. On obtient alors $\mathsf{erase}(\mathsf{merge}(\mathsf{PG}(\mu)))$ de la Figure 5.12.

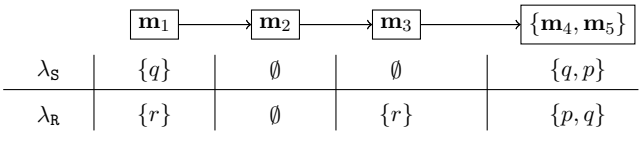


FIGURE 5.12 – $\mathsf{erase}(\mathsf{merge}(\mathsf{PG}(\mu)))$

✢ La dernière étape, la fonction $\mathsf{sweep}(\cdot)$ supprime les sommets ne comportant plus d'informations sur les processus. En l'occurrence, elle nous permet ici de supprimer le sommet $\mathbf{m}_2$. On obtient enfin l'abstraction $\alpha(\mathsf{PG}(\mu))$ de la Figure 5.13.

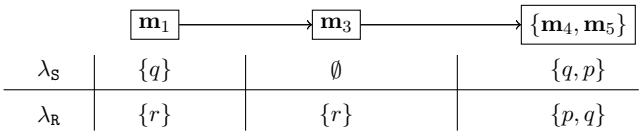


FIGURE 5.13 – $\mathsf{sweep}(\mathsf{erase}(\mathsf{merge}(\mathsf{PG}(\mu))))$

Nous pouvons alors démontrer que si l'abstraction du graphe de dépendances ne contient qu'un sommet, nous avons bien à faire à un échange premier.

Lemme 5.3.2. $\mathsf{GD}(\mu)$ *est fortement connecté si et seulement si* $\alpha(\mathsf{PG}(\mu))$ *est un graphe à un seul sommet.*

Démonstration.

Par définition de α, et en particulier de $\mathsf{merge}(\cdot)$, un sommet de $\alpha(\mathsf{PG}(\mu))$ correspond à une composante fortement connexe de $\mathsf{GD}(\mu)$.

□

Par construction, pour tout processus p, et pour tout $\mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}$, il y a au plus deux sommets $\mathbf{m}, \mathbf{m}'$ de $\alpha(\mathsf{PG}(\mu))$ tels que $p \in \lambda_{\mathtt{X}}(\mathbf{m})$ et $p \in \lambda_{\mathtt{X}}(\mathbf{m}')$. À partir de cela, on peut alors déduire que $\alpha(\mathsf{PG}(\mu))$ a au plus $2\times \mid \mathbb{P} \mid$ sommets, et donc, par conséquent :

Lemme 5.3.3. $\sharp\{\alpha(\mathsf{PG}(\mu)) \mid \mu \textit{ est un échange}\} \leq 2^{9|\mathbb{P}|^2}$

Démonstration.

Soit $n \geq 0$ fixé et donnons une borne supérieure au nombre de P-graphes à n sommets. Premièrement, il y a $2^{n(n-1)}$ choix différents possibles pour les arcs. Par construction (en particulier, par définition de la fonction $\mathsf{erase}(\cdot)$), on a :

$$\forall p \in \mathbb{P}, \forall \mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}, \sharp\{v \mid p \in \lambda_{\mathtt{X}}(\mathbf{m})\} \leq 2 \tag{5.1}$$

Un choix pour la fonction λ est donc le choix, pour chaque p, d'au plus deux sommets tels que $p \in \lambda_{\mathtt{S}}(m)$ et au plus deux autres sommets m tels que $p \in \lambda_{\mathtt{R}}(m)$. Donc il y a au plus n^4 choix différents pour chaque p, et au plus $n^{4|\mathbb{P}|}$ choix différents pour λ. Pour résumer, il a moins de $2^{n^2} n^{4|\mathbb{P}|}$ P-graphes avec n sommets. Maintenant, d'après (5.1), il y a au plus $2|\mathbb{P}|$ sommets dans un P-graphe, donc le nombre de P-graphes est borné par

$$\textstyle\sum_{n=1}^{2|\mathbb{P}|} 2^{n^2} n^{4|\mathbb{P}|} \leq (2|\mathbb{P}|) 2^{(2|\mathbb{P}|)^2} (2|\mathbb{P}|)^{4|\mathbb{P}|} \leq 2^{|\mathbb{P}|^2} 2^{4|\mathbb{P}|^2} 2^{4|\mathbb{P}|^2} \leq 2^{9|\mathbb{P}|^2}$$

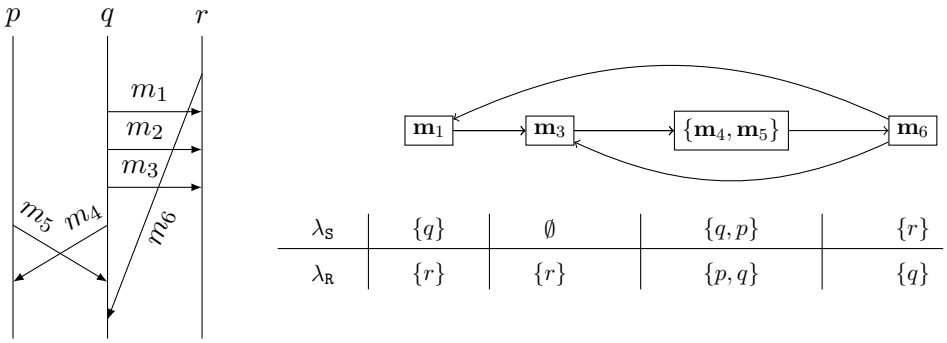
□

Ainsi, il existe un nombre fini d'abstractions $\alpha(\mathsf{PG}(\mu))$. Ceci nous permet d'affirmer qu'il existe un automate fini capable de reconnaître les échanges premiers, et la dernière étape consiste donc à le construire. Celui-ci aura pour états des abstractions de graphe de dépendances et les

transitions seront des lettres de Σ, c'est-à-dire l'envoi et la réception d'un message ou seulement l'envoi. Les états finaux seront alors les abstractions de graphe de dépendances formées d'un seul sommet, ce qui assurera que le graphe abstrait observé n'est formé que d'une composante fortement connexe.

On nomme cet automate $\mathtt{P} = (L_\mathtt{P}, \delta_\mathtt{P}, \ell^0_\mathtt{P}, F_\mathtt{P})$ et on le définit comme suit :

- $L_\mathtt{P} = \{\alpha(\mathsf{PG}(\mu)) \mid \mu \text{ est un échange}\}$,
- $\ell^0_\mathtt{P} = (\emptyset, \emptyset, \emptyset, \emptyset)$ le graphe vide,
- $F_\mathtt{P} = \{(V, E, \lambda_\mathtt{S}, \lambda_\mathtt{R}) \mid \mid V \mid = 1\}$,
- la relation de transition $\delta_\mathtt{P}$ est définie ainsi : il existe $G \xrightarrow{\dagger m^{p\to q}} G'$ avec $G = (V, E, \lambda_\mathtt{S}, \lambda_\mathtt{R})$ et $G' = (V', E', \lambda'_\mathtt{S}, \lambda'_\mathtt{R})$ si
  - $V' = V \uplus \{\mathbf{m}\}$,
  - $\lambda'_\mathtt{S}(\mathbf{m}) = \{p\}$,
  - si $\dagger = !?$ alors $\lambda'_\mathtt{R}(\mathbf{m}) = \{q\}$, sinon $\lambda'_\mathtt{R}(\mathbf{m}) = \emptyset$,
  - pour tout $\mathbf{m}'$, et pour tout $\mathtt{X} \in \{\mathtt{S}, \mathtt{R}\}$, $\lambda'_\mathtt{X}(\mathbf{m}') = \lambda_\mathtt{X}(\mathbf{m}')$,
  - $E' = E \cup \{(\mathbf{m}', \mathbf{m}) \mid p \in \lambda_\mathtt{S}(\mathbf{m}')\} \cup \{(\mathbf{m}, \mathbf{m}') \mid p \in \lambda_\mathtt{R}(\mathbf{m}')\} \cup \begin{cases} \{(\mathbf{m}', \mathbf{m}) \mid q \in \lambda_\mathtt{S}(\mathbf{m}') \cup \lambda_\mathtt{R}(\mathbf{m}')\} & \text{si } \dagger = !? \\ \emptyset & \text{si } \dagger = ! \end{cases}$



(a) **(b)**

FIGURE 5.14 – MSC μ avec le message m_6 (a) et le P-graphe associé (b)

Exemple 5.3.2 – Considérons l'abstraction $G = \alpha(\mathsf{PG}(\mu))$ de la Figure 5.13 comme un état de l'automate P et supposons que la transition disponible est la suivante $\xrightarrow{!?m_6^{r\to q}}$. Si l'on ajoute ce message m_6 après la séquence précédente, on obtient le MSC de la Figure 5.14.a. L'état d'arrivée correspond à l'abstraction $\alpha(G')$ où G' représenté en Figure 5.14.b. contient le sommet $\mathbf{m}_6$ en plus, ainsi que les arcs associés. Nous pouvons en effet ajouter les arcs suivants :

- $\mathbf{m}_6 \to \mathbf{m}_1$ car $r \in \lambda_\mathtt{R}(\mathbf{m}_1)$

- $\mathbf{m}_6 \to \mathbf{m}_3$ car $r \in \lambda_{\mathtt{R}}(\mathbf{m}_3)$
- $\mathbf{m}_1 \to \mathbf{m}_6$ car $q \in \lambda_{\mathtt{S}}(\mathbf{m}_1) \cup \lambda_{\mathtt{R}}(\mathbf{m}_1)$
- $\{\mathbf{m}_4, \mathbf{m}_5\} \to \mathbf{m}_6$ car $q \in \lambda_{\mathtt{S}}(\{\mathbf{m}_4, \mathbf{m}_5\}) \cup \lambda_{\mathtt{R}}(\{\mathbf{m}_4, \mathbf{m}_5\})$

On constate dès la première étape que $\mathsf{merge}(G')$, et donc $\alpha(G')$, rassemble tous les sommets dans le même sommet et forme alors un état final, contenant un seul et unique sommet.

Montrons alors que la construction effectuée dans l'automate P est correcte.

Lemme 5.3.4. *$\alpha(\mathsf{PG}(msc(w))) \xrightarrow{\dagger m^{p\to q}} G'$ si et seulement si $G' = \alpha(\mathsf{PG}(msc(w \cdot \dagger m^{p\to q})))$.*

Avant de prouver le Lemme 5.3.4, nous avons besoin d'introduire quelques notations et observations. Soit $G = (V, E, \lambda_{\mathtt{S}}, \lambda_{\mathtt{R}})$ un P-graphe. Un sommet $\mathbf{m} \in V$ est $\mathtt{X}$-*couvert* si pour tout $p \in \lambda_{\mathtt{X}}(\mathbf{m})$, p est $\mathtt{X}$-redondant. On dira aussi que $\mathbf{m} \in V$ est couvert s'il est $\mathtt{S}$-couvert et $\mathtt{R}$-couvert. Une abstraction partielle de G est un graphe $G' = (V', E', \lambda_{\mathtt{S}}, \lambda_{\mathtt{R}})$ tel que

- $V' = \{V_1, \cdots, V_n\}$ où chaque V_i est une composante fortement connexe (pas nécessairement maximale), tous les V_i sont disjoints et pour tout $\mathbf{m} \in V \setminus \bigcup_{i=1}^{n} V_i$, $\mathbf{m}$ est couvert,
- pour tout $i, j, (V_i, V_j) \in E'$ si et seulement si $(\mathbf{m}, \mathbf{m}') \in E$ pour certains $\mathbf{m} \in V_i$ et $\mathbf{m}' \in V_j$,
- pour tout $i, X, \lambda_X(V_i) = \bigcup_{v \in V_i} \lambda_X(\mathbf{m})$.

Intuitivement, G' est une abstraction partielle de G si elle résulte d'une "application partielle" des fonctions $\mathsf{merge}(\cdot)$, $\mathsf{erase}(\cdot)$ et $\mathsf{sweep}(\cdot)$: certains sommets d'une même composante fortement connexe sont fusionnés, mais pas nécessairement tous, certaines étiquettes sont effacées, mais pas nécessairement toutes, et certains sommets seront supprimés, mais pas nécessairement tous. À partir de cette observation, on peut alors déduire que : si G' est une abstraction partielle de G, alors $\alpha(G') = \alpha(G)$.
Démonstration.

Soient $w \in \Sigma^*$ et $\dagger m^{p\to q}$ donnés. Soit $G_1 = \mathsf{PG}(msc(w))$ et $G_2 = \mathsf{PG}(msc(w \cdot \dagger m^{p\to q}))$ et comparons alors G_1 et G_2. D'abord, il existe un sommet supplémentaire $\mathbf{m}_0$ dans G_2 qui représente $\dagger m^{p\to q}$, avec $\lambda_{\mathtt{S}}(\mathbf{m}_0) = \{p\}$ et soit $\lambda_{\mathtt{R}}(\mathbf{m}_0) = \{q\}$ (si $\dagger =!?$) soit $\lambda_{\mathtt{R}}(\mathbf{m}_0) = \emptyset$ (si $\dagger =?$).

Maintenant, considérons les arcs supplémentaires. Évidemment, ces arcs ont $\mathbf{m}_0$ comme source ou comme destination. Commençons par les arcs avec $\mathbf{m}_0$ comme destination. L'envoi de $\mathbf{m}_0$ se produit après tous les évènements de p, donc pour tout $\mathbf{m} \neq \mathbf{m}_0$ tel que $p \in \lambda_{\mathtt{S}}(\mathbf{m})$, $(\mathbf{m}, \mathbf{m}_0) \in E_2$. Dans le cas où $\dagger =!?$, l'évènement de réception de $\mathbf{m}_0$ doit aussi avoir lieu après tous les envois et réceptions de q, donc pour tout $\mathbf{m} \neq \mathbf{m}_0$ tel que $q \in \lambda_{\mathtt{S}}(\mathbf{m}) \cup \lambda_{\mathtt{R}}(\mathbf{m})$, $(\mathbf{m}, \mathbf{m}_0) \in E_2$. Il n'y a pas d'autres arcs entrant dans $\mathbf{m}_0$.

Passons maintenant aux arcs sortants de $\mathbf{m}_0$. L'envoi de $\mathbf{m}_0$ a lieu avant tous les envois de p, donc pour tout $\mathbf{m} \neq \mathbf{m}_0$ tel que $p \in \lambda_{\mathtt{R}}(\mathbf{m})$, $(\mathbf{m}_0, \mathbf{m}) \in E_2$.

Pour résumer, nous avons :

$$\begin{array}{rcl} E_2 & = & E_1 \cup \{(\mathbf{m}, \mathbf{m}_0) \mid p \in \lambda_{\mathtt{S}}(\mathbf{m})\} \\ & \cup & \{(\mathbf{m}_0, \mathbf{m}) \mid p \in \lambda_{\mathtt{R}}(\mathbf{m})\} \\ & \cup & \begin{cases} \{(\mathbf{m}, \mathbf{m}_0) \mid q \in \lambda_{\mathtt{S}}(\mathbf{m}) \cup \lambda_{\mathtt{R}}(\mathbf{m})\} & \text{si } \dagger = !? \\ \emptyset & \text{si } \dagger = ! \end{cases} \end{array}$$

Observons à présent que les règles qui ajoutent des sommets et des arcs pour passer de G_1 à G_2 sont exactement identiques aux règles pour passer de G à G' dans la définition de $\delta_g(G, \dagger m^{p \to q})$. Supposons que $G = \alpha(G_1) = \alpha(\mathsf{PG}(msc(w))$. Alors, G' est une abstraction partielle de G_2. Et donc, au vu de la discussion apportée ci-dessus,

$$\alpha(G') = \alpha(G_2).$$

Ainsi, par définition de $\delta_{\mathtt{P}}$, $G \xrightarrow{\dagger m^{p \to q}} \alpha(G')$. Pour résumer,

$$G \xrightarrow{\dagger m^{p \to q}} \alpha(G_2) = \alpha(G_2) = \alpha(\mathsf{PG}(msc((w \cdot \dagger m^{p \to q}))).$$

□

Comme l'automate P est correct, il existe donc bien un automate régulier acceptant le langage des échanges premiers.

Théorème 5.3.5. *Il existe un automate fini déterministe avec au plus* $2^{9|\mathbb{P}|^2}$ *états tel que* $\mathcal{L}(\mathtt{P}) = \{w \in \Sigma^* \mid msc(w)$ *est premier*$\}$.

Démonstration.

Soit l'automate P l'automate défini dans la Section 5.3. Alors, par le Lemme 5.3.3, P est un automate fini déterministe avec au plus $2^{9|\mathbb{P}|^2}$ états. De plus, par le Lemme 5.3.4, pour tout w, la construction est correct et complète et w est accepté si et seulement si $\mathsf{PG}(msc(w))$ est un graphe à un seul sommet. Par le Lemme 5.3.2, c'est équivalent au fait que $msc(w)$ est premier. L'automate P reconnaît alors tous les échanges premiers du système ce qui conclut cette preuve.

□

5.4 Le calcul de k

Dans les sections précédentes, nous avons vu qu'un moyen de calculer le degré de synchronisabilité $\mathsf{sd}(\mathcal{S})$ est de calculer la longueur k du plus grand échange premier et accessible (Lemme 5.1.1). Pour tout mot $w \in \Sigma^*$, on associe un MSC $msc(w)$ et on a montré que pour chaque MSC μ accessible, il existe un mot $w \in \Sigma^*$ tel que $\mu = msc(w)$ (Lemme 5.1.2). Nous pouvons alors déduire que k correspond à la longueur du plus long mot de $\mathcal{L}_a \cap \mathcal{L}_p$ si $\mathcal{L}_a \cap \mathcal{L}_p$ est fini, sinon $k = \infty$. Dans la Section 5.2, nous avons montré que $\mathcal{L}_a$ est un langage régulier effectif. La Section 5.3 a montré que le langage $\mathcal{L}_p$ est lui aussi régulier. On déduit alors que $\mathcal{L}_a \cap \mathcal{L}_p$ est également un langage régulier, et que k est calculable, puisque la finitude et la longueur du mot le plus long d'un langage régulier sont calculables. Avec une analyse attentive des automates en jeu, une borne supérieure pour la valeur de k peut être donnée.

Théorème 5.4.1. $\mathsf{sd}(\mathcal{S})$ *est calculable, et si* $\mathsf{sd}(\mathcal{S}) < \infty$ *alors* $\mathsf{sd}(\mathcal{S}) <\mid \mathcal{S} \mid^2 2^{11|\mathbb{P}|^2}$*, où* $\mid \mathcal{S} \mid$ *est le nombre d'états de contrôle globaux et* $\mid \mathbb{P} \mid$ *le nombre de processus de* $\mathcal{S}$*.*

Démonstration.

Le fait que $\mathsf{sd}(\mathcal{S})$ est calculable est expliquée dans la Section 5.4. Nous ne prouverons ici seulement le fait que, si $k < \infty$ alors $k < |\mathcal{S}|^2 2^{11|\mathbb{P}|^2}$. k est la longueur du plus long mot de $\mathcal{L}_a \cap \mathcal{L}_p$. Par le Théorème 5.2.4, $\mathcal{L}_a$ est régulier et accepté par un automate fini. Par le Théorème 5.3.5, il existe un automate $\mathtt{P}$ tel que $\mathcal{L}(\mathtt{P}) = \mathcal{L}_p$. Ainsi, nous avons besoin d'une borne sur la longueur du plus long mot de

$$\mathcal{L}(\mathtt{SR}(\vec{\ell}, \overrightarrow{mid}, \vec{\ell}')) \cap \mathtt{MR}(\mathcal{B}, \mathcal{B}') \cap \mathcal{L}(\mathtt{P})$$

en supposant que ce langage est fini pour $\vec{\ell}, \overrightarrow{mid}, \vec{\ell}', \mathcal{B}, \mathcal{B}'$. Cette borne est donnée par le nombre d'états de chacun de ces automates qui acceptent ce langage (puisqu'un mot plus long nécessiterait une boucle dans l'automate, et le langage ne serait donc pas fini). Ce langage est reconnu par un automate qui est le produit des automates

$\mathtt{SR}(\vec{\ell}, \overrightarrow{mid}, \vec{\ell}')$, $\mathtt{MR}(\mathcal{B}, \mathcal{B}')$ et $\mathtt{P}$ et donc son nombre d'états est borné par

$$|\mathtt{SR}(\vec{\ell}, \overrightarrow{mid}, \vec{\ell}')| \times |\mathtt{MR}(\mathcal{B}, \mathcal{B}')| \times |\mathtt{P}|$$

Par définition de $\mathtt{SR}$, $L_{\mathtt{SR}} = L_{\mathcal{S}}^2$, donc $\mid \mathtt{SR} \mid \leq \mid L_{\mathcal{S}}^2 \mid$ (qui peut aussi être noté $\mid \mathcal{S} \mid^2$).

Par définition de $\mathtt{MR}$, $L_{\mathtt{MR}} = \mathbb{B} = (2^{\mathbb{P}} \times 2^{\mathbb{P}})^{\mathbb{P}}$, donc $\mid \mathtt{MR} \mid \leq 2^{2|\mathbb{P}|^2}$.

Enfin, par le Théorème 5.3.5, $\mid \mathtt{P} \mid \leq 2^{9|\mathbb{P}|^2}$. Tout cela réunit, l'on obtient

$$\begin{aligned} k &\leq \mid \mathcal{S} \mid^2 2^{2|\mathbb{P}|^2} 2^{9|\mathbb{P}|^2} \\ k &\leq \mid \mathcal{S} \mid^2 2^{11|\mathbb{P}|^2} \end{aligned}$$

□

On peut alors établir la décidabilité pour un système communiquant en boîte aux lettres.

Théorème 5.4.2. *Le problème suivant est décidable : pour un système donné* $\mathcal{S}$*, existe-t-il un* k *tel que* $\mathcal{S}$ *est* k*-synchronisable ?*

5.5 Conclusion

Ce chapitre établit la décidabilité de la k-synchronisabilité pour un système sans connaître le paramètre k à l'avance. En particulier, nous avons démontré que l'on est capable de caractériser le plus petit k tel qu'un système donné est k-synchronisable, voire même d'identifier un système où un tel k ne peut exister. La décidabilité consiste en la recherche de ce k qui correspond donc à la taille du plus grand échange accessible et premier. La preuve s'appuie précisément sur la régularité du langage de séquences d'actions représentant des échanges accessibles et de celui des séquences d'actions décrivant des échanges premiers, ce qui est rendu possible grâce à la sémantique boîte aux lettres. En pair à pair, le problème reste ouvert.

CHAPITRE 6

Thème et variations

La k-synchronisabilité permet, comme nous avons pu le voir dans les chapitres précédents, de décider de l'accessibilité d'un état de contrôle. Nous sommes également capables de savoir si le système étudié est k-synchronisable pour un k donné en paramètre, ou même sans k particulier. Cependant, certaines critiques peuvent être faites à la définition de la k-synchronisabilité.

Ce chapitre s'appliquera, dans un premier temps, à les expliciter. Nous verrons des incohérences entre l'intuition qui pourrait se dégager de cette définition et des cas de marge que l'on voudrait alors éviter. La deuxième section définira des variations à la définition de k-synchronisabilité. La première résoudra drastiquement les incohérences alors évoquées, tandis que la seconde nous permettra de les corriger avec une classe plus large. La troisième section comparera ces nouvelles définitions à celles déjà existantes, la k-synchronisabilité mais également les bornes existentielles et universelles. Chaque cas possible sera représenté par un exemple. Enfin, nous terminerons ce chapitre par les preuves de décidabilité des problèmes d'accessibilité et d'appartenance pour les classes définies plus tôt.

6.1 Critiques : des cas inattendus

La k-synchronisabilité peut révéler des cas non intuitifs. Ceux-ci concernent deux sujets : la division en k-échanges et les bornes sur les canaux. Ce genre d'incohérences ne peut avoir lieu que dans une communication en boîte aux lettres. Elles sont en effet dues aux messages non couplés stagnant dans les canaux.

Plus précisément, la division en k-échanges peut être trompeuse et ne correspond pas toujours à une exécution. Il se peut que, pour obtenir une linéarisation mb-réalisable d'un MSC k-synchrone, il y est l'obligation d'intervertir des actions de k-échanges différents. Si un message non couplé mais apparemment indépendant peut être mis dans un premier k-échange, il se peut qu'un message non couplé, appartenant à un k-échange subséquent, doit être envoyé avant pour être effectivement lu. Nous pouvons constater ceci dans l'exemple qui suit.

Exemple 6.1.1 – Regardons un premier exemple pour expliciter cela. Le MSC de la Figure 6.1 est 2-synchrone : on y voit d'ailleurs la division correspondante. Cependant, cette séquence de k-échanges n'admet pas d'exécution mb-réalisable. En effet, dans une linéarisation mb-réalisable, l'envoi du message m_3 doit précéder l'envoi du message m_1. Ainsi, le découpage ne correspond pas à une exécution mb-réalisable.

Ce cas donne à réfléchir sur la définition et le sens des k-échanges : ils sont intuitivement supposés rassembler des messages dépendants les uns des autres, mais comme nous venons que de le constater, cela n'est pas toujours le cas. Pour y remédier et avoir un découpage cohérent avec

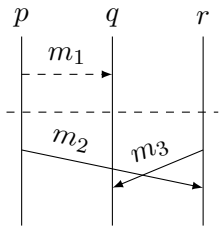


FIGURE 6.1 – Un MSC 2-synchrone

les comportements possibles du système, on imposera que la séquence de k-échanges constituant la division d'un MSC k-synchrone corresponde également à une exécution du système.

Une seconde critique peut être faite, cette fois-ci à propos des bornes sur les canaux. Comme tout MSC peut être divisé en phase de k messages, il serait légitime de penser qu'un maximum de k messages peut être contenu dans chaque canal. Or, ce n'est pas le cas. Comme vu précédemment, l'ordre des actions pour obtenir une linéarisation mb-réalisable n'est pas toujours lié au découpage en k-échanges du MSC concerné. Cela implique le stockage de plus de messages qu'imaginé. Regardons un nouvel exemple pour ce cas.

Exemple 6.1.2 – Supposons un MSC tel que celui de la Figure 6.2. Le nombre d'occurrences du message m_2 est inconnu. Comme on le constate visuellement, cet MSC est 1-synchrone. Cependant, pour qu'une linéarisation soit une exécution mb-réalisable, le message m_3 doit être envoyé avant le message m_1. Or, pour envoyer le message m_3, les messages m_2 doivent être envoyés. Ils devront néanmoins attendre l'envoi du message m_1 pour être lus. Si l'on considère deux occurrences du message m_2, le MSC est alors $\exists$-2-mb-borné : les deux messages m_2 devront être stockés dans c_p en attendant l'envoi de m_3 et de m_1. On peut alors étendre ceci à tous les MSC de cette forme, chacun est $\exists$-k-mb-borné avec k le nombre d'occurrences de m_2. Le système associé n'est alors pas $\exists$-k-mb-borné car il n'existe pas un k_{max} tel que tous les MSC soient $\exists$-k_{max}-mb-borné.

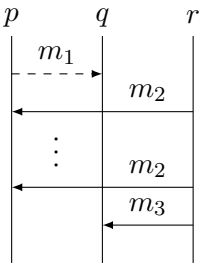


FIGURE 6.2 – Un MSC 1-synchrone

Il se trouve que ce résultat inattendu peut être changé grâce à la solution évoquée plus tôt : forcer la division à correspondre à une exécution du système. Ainsi, un système sera k-synchronisable si tout MSC peut être divisé en k-échanges, avec un k fixe. Si les k-échanges peuvent être de taille arbitrairement grande, un nombre non borné de messages pourront être stockés avant d'être lus et le système ne sera donc pas $\exists$-k-mb-borné.

Pour résoudre les incohérences soulevées, nous définissons dans la prochaine section deux nouvelles classes de systèmes.

6.2 Des variations de la k-synchronisabilité

Ainsi, dans cette section, nous définirons de nouvelles définitions permettant d'éviter les cas de marges présentés précédemment. Pour cela, nous ajoutons la contrainte suivante : la division en k-échanges doit correspondre à une exécution mb-réalisable. Avec l'ajout de cette contrainte, nous obtenons la classe des systèmes fortement synchronisables, noté k_F-synchronisables. Nous y définirons une caractérisation graphique correspondante.

Dans un second temps, nous verrons que tout en conservant cette contrainte, nous pouvons en relâcher une autre : l'ordre sur les envois et les réceptions dans un k-échange. En autorisant l'alternance des types d'actions dans un k-échange, cela nous permettra de capturer de nouveaux systèmes tout en gardant un découpage cohérent et une borne sur les canaux. Nous apporterons pour cette définition également une caractérisation graphique correspondante.

6.2.1 La k-synchronisabilité forte

Nous commençons donc par la définition de la k-synchronisabilité forte, noté k_F-synchronisabilité. Afin de pouvoir comparer le comportement de cette classe dans les différents types de commmunication, nous définirons la k_F-synchronisabilité en boîte aux lettres ainsi qu'en pair à pair.

Notre objectif est donc de capturer les systèmes qui sont k-synchronisables mais également $\exists$-k-com-borné (Définition 3.3.4) et dont le découpage de chaque MSC k_F-synchrone soit cohérent avec une exécution du système. Pour cela, on reconnaît un MSC fortement k-synchronisable, noté k_F-synchronisable, par la présence d'une linéarisation divisible en k-échanges qui est une exécution du système. Autrement dit, on attend qu'une linéarisation du MSC soit une exécution k-com-bornée et divisible en k-échanges.

On peut alors définir un k_F-échange, un MSC k_F-synchrone et une exécution k_F-synchronisable formellement.

Définition 6.2.1 (k_F-échange)**.** Une séquence d'action $e = a_1 \cdots a_n$ est un k_F-échange si $e \in \mathtt{S}^{\leq k} \cdot \mathtt{R}^{\leq}$ telle que pour tout $j \in [1..n]$ tel que $a_j \in \mathtt{R}$, il existe $i \in [1..j[$ tel que $a_i \in \mathtt{S}$ et $a_i \mapsto a_j$.

Définition 6.2.2 (MSC k_F-synchrone)**.** Un MSC est k_F-synchrone en boîte aux lettres, respectivement en pair à pair, s'il existe une linéarisation $e = e_1 \cdots e_n$ de μ telle que :

- pour tout $i \in [1..n]$, e_i est un k_F-échange,
- e est mb-réalisable, respectivement, pp-réalisable.

Une exécution e est fortement k-synchronisable, noté k_F-synchronisable, si $msc(e)$ est k_F-synchrone.

On note $\textsc{Msc}_k^F(\mathcal{S})$ l'ensemble des MSC k_F-synchrones d'un système $\mathcal{S}$. On parlera de k_F-échanges si ceux-ci peuvent être concaténés et mener à une exécution k_F-synchronisable.

Définition 6.2.3 (Système k_F-synchronisable)**.** Un système $\mathcal{S}$ est k_F-synchronisable si $\textsc{Msc}(\mathcal{S}) = \textsc{Msc}_k^F(\mathcal{S})$.

Les exemples suivants doivent être considérés dans le contexte d'une communication en boîte aux lettres.

Exemple 6.2.1 – Le MSC de la Figure 6.3.a est 1-synchrone mais 3_F-synchrone. En effet, la linéarisation $e = \mathtt{s}(p,q,m_1) \cdot \mathtt{s}(r,p,m_2) \cdot \mathtt{r}(r,p,m_2) \cdot \mathtt{s}(r,q,m_3) \cdot \mathtt{r}(r,q,m_3)$ est divisible en 1-échanges mais n'est pas une exécution mb-réalisable : le message m_1 non couplé est envoyé avant le message m_3 couplé donc le message m_3 ne peut pas être lu, le message m_1 étant en tête de file dans le canal c_q. Si l'on souhaite une linéarisation qui soit une exécution, on aura par exemple $e' = \mathtt{s}(r,p,m_2) \cdot \mathtt{s}(r,q,m_3) \cdot \mathtt{s}(p,q,m_1) \cdot \mathtt{r}(r,p,m_2) \cdot \mathtt{r}(r,q,m_3)$ qui elle n'est pas divisible et constitue un 3_F-échange.

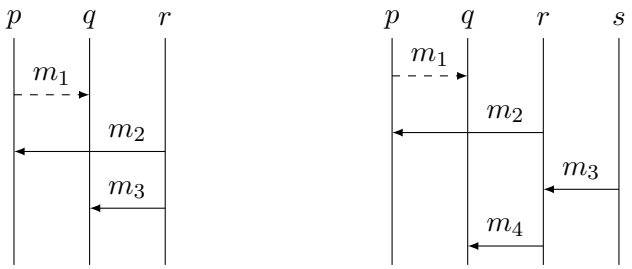


FIGURE 6.3 – Un MSC 3_F-synchrone (a) et un MSC pas k_F-synchrone (b)

Le MSC de la Figure 6.3.b est lui 1-synchrone mais n'est pas k_F-synchrone pour aucun k. La linéarisation $e = \mathtt{s}(p,q,m_1) \cdot \mathtt{s}(r,p,m_2) \cdot \mathtt{r}(r,p,m_2) \cdot \mathtt{s}(s,r,m_3) \cdot \mathtt{r}(s,r,m_3) \cdot \mathtt{s}(r,q,m_4) \cdot \mathtt{r}(r,q,m_4)$ est en effet divisible en 1-échanges mais n'est pas une exécution mb-réalisable, pour la même raison que précédemment : le message m_1 non couplé empêche le message m_4, couplé, d'être lu. Cependant les linéarisations correspondant à des exécutions mb-réalisable présentent un autre problème. Prenons $e' = \mathtt{s}(r,p,m_2) \cdot \mathtt{s}(p,q,m_3) \cdot \mathtt{s}(s,r,m_1) \cdot \mathtt{r}(s,r,m_3) \cdot \mathtt{s}(r,q,m_4) \cdot \mathtt{r}(r,q,m_4) \cdot \mathtt{r}(r,p,m_2)$, e' est une exécution mb-réalisable, mais n'est pas divisible en k-échanges : nous ne pouvons pas réorganiser les actions de façon à avoir tous les envois suivis des réceptions.

On peut faire une première constatation évidente sur les systèmes k_F-synchronisables : si un système est k_F-synchronisable alors il est également k-synchronisable.

Lemme 6.2.1. *Soit μ un MSC k_F-synchrone, il existe un $k' \leq k$ tel que μ est k'-synchrone.*

Démonstration.

Par définition d'un MSC k_F-synchrone μ, on sait qu'il existe une linéarisation $e = e_1 \cdots e_n$ de μ telle que :

- pour tout $i \in [1..n]$, $e_i \in \mathtt{S}^{\leq k} \cdot \mathtt{R}^{\leq k}$, ce qui correspond à la première condition de la définition pour qu'un MSC k-synchrone,
- e est mb-réalisable, ou pp-réalisable selon le type de communication du système, donc $msc(e) = \mu$ est réalisable, ce qui correspond à la deuxième condition de la définition pour qu'un MSC k-synchrone,

- ✢ pour tout j, j' tels que $a_j \vdash a_{j'}$ appartiennent à e, il existe i tel que $a_j, a_{j'}$ appartiennent à e_i, ce qui correspond à la dernière condition de la définition pour qu'un MSC k-synchrone.

Donc, un MSC k_F-synchrone est k-synchrone. μ peut être k'-synchrone pour un $k' < k$ si une autre linéarisation, qui n'est pas une exécution, peut être divisible en k-échanges.

□

Dans le cas d'une communication en pair à pair, les messages non couplés n'impliquent pas d'organisation particulière pour obtenir une exécution pp-réalisable : un message non couplé empêche seulement les messages envoyés par le même expéditeur au même destinataire d'être lus s'ils sont envoyés après lui, ce qui est alors directement induit par le MSC et non pas dépendant de la linéarisation choisie. Ainsi, si une exécution pp-réalisable est représentée par un MSC, toutes les linéarisations de ce MSC seront pp-réalisables. Finalement, on peut prouver l'égalité des classes des systèmes k-synchronisables et celle des systèmes k_F-synchronisables.

Lemme 6.2.2. *Soit $\mathcal{S}$ un système où* $\mathsf{com} = \mathsf{pp}$*, et $\mu \in \textsc{Msc}(\mathcal{S})$ un MSC. μ est k_F-synchrone si et seulement si μ est k-synchrone.*

Démonstration.

$\Rightarrow$ Par le Lemme 6.2.1.

$\Leftarrow$ Soit μ un MSC μ est k-synchrone. Donc, il existe une linéarisation $e = e_1 \cdots e_n$ tel que e_i est un k-échange, $1 \leq i \leq n$. μ est k-synchrone donc pp-réalisable. Donc, il n'existe pas $j < j'$ tel que $a_j = \mathsf{s}(p, q, m), a'_j = \mathsf{s}(p, q, m') \in e$ avec a_j non couplé et a'_j couplé. Donc e est pp-réalisable. Donc μ est k_F-synchrone.

□

Enfin, nous pourrons également prouver plus tard que tout système k_F-synchronisable est également $\exists$-k-com-borné.

6.2.2 Caractérisations graphiques de la k_F-synchronisabilité

À l'instar des MSC k-synchrones, nous pouvons caractériser graphiquement la division des MSC k_F-synchrones. Deux cas se distinguent alors : dans la communication en pair à pair, nous utiliserons le graphe de dépendances tout comme dans les systèmes k-synchronisables, alors que dans la communication en boîte aux lettres, nous utiliserons le graphe de dépendances étendu.

Commençons par le cas de la communication en pair à pair, pour laquelle nous pouvons utiliser les propriétés d'un système k-synchronisable. Nous avons prouvé qu'un MSC est k-synchronisable si et seulement s'il est k_F-synchronisable. Ainsi, le découpage en k_F-échanges d'un MSC est le même que le découpage en k-échanges dans une communication en pair à pair, et nécessite de regarder les composantes fortement connexes dans le graphe de dépendances.

Théorème 6.2.3. *Soient $\mathcal{S}$ un système avec* $\mathsf{com} = \mathsf{pp}$ *et $\mu \in \textsc{Msc}(\mathcal{S})$. μ est k_F-synchrone si et seulement si toute composante fortement connexe de graphe de dépendances $\mathsf{GD}(\mu)$ est de taille inférieure ou égale à k et ne contient pas d'arc RS.*

Démonstration.

Soit $\mu \in \text{MSC}(\mathcal{S})$ où com = pp.

$\Rightarrow$ Si μ est k_F-synchrone alors, d'après le Lemme 6.2.2, μ est k-synchrone. Ainsi, d'après le Théorème 4.1.2, toute composante fortement connexe dans le $\mathsf{GD}(\mu)$ est de taille inférieure ou égale à k et ne contient pas d'arc RS.

$\Leftarrow$ Si toute composante fortement connexe de $\mathsf{GD}(\mu)$ est de taille inférieure ou égale à k et ne contient pas d'arc RS, alors, d'après le Théorème 4.1.2, μ est k-synchrone. D'après le Lemme 6.2.2, μ est alors k_F-synchrone.

□

Dans une communication en boîte aux lettres, le découpage n'est pas toujours identique entre k-échanges et k_F-échanges. En effet, comme on a pu le voir dans l'Exemple 6.2.1, les messages non couplés peuvent nécessiter une organisation particulière pour obtenir une exécution mb-réalisable.

Nous avons vu que le graphe de dépendances étendu permet d'identifier l'ordre partiel nécessaire pour être mb-réalisable. Ainsi, il pourra ici nous permettre de construire les k_F-échanges en regroupant les messages dépendants les uns des autres dans les différentes composantes fortement connexes.

Tout comme précédemment, les composantes fortement connexes ne devront pas contenir d'arc RS pour assurer que l'on puisse faire tous les envois avant les réceptions et ne devront pas contenir plus de k messages pour que le MSC associé soit k_F-synchrone.

Théorème 6.2.4. *Soient $\mathcal{S}$ un système avec* com = mb *et $\mu \in \text{MSC}(\mathcal{S})$. μ est k_F-synchrone si et seulement si toute composante fortement connexe du graphe de dépendances étendu $\mathsf{GDE}(\mu)$ est de taille inférieure ou égale à k et ne contient pas d'arc RS.*

Démonstration.

Soit $\mu \in \text{MSC}(\mathcal{S})$ où com = mb.

$\Rightarrow$ Si μ est k_F-synchrone, alors $\exists e, e = e_1 \cdots e_n$ où chaque e_i est un k_F-échange. Pour chaque sommet $\mathbf{m}$ du graphe de dépendances étendu $\mathsf{GDE}(e)$, il y a exactement un index $\iota(\mathbf{m}) \in [1..n]$ tel que $\mathbf{m} \subseteq e_{\iota(\mathbf{m})}$. Maintenant, observons que, s'il y a un arc de $\mathbf{m}$ à $\mathbf{m}'$ dans le graphe de dépendances étendu, une des actions de $\mathbf{m}$ doit avoir lieu avant une action de $\mathbf{m}'$ pour obtenir une exécution mb-réalisable, c'est-à-dire, $\iota(\mathbf{m}) \leq \iota(\mathbf{m}')$. Donc, si $\mathbf{m}, \mathbf{m}'$ sont dans la même composante fortement connexe, alors $\iota(\mathbf{m}) = \iota(\mathbf{m}')$ et ils doivent apparaître dans le même k_F-échange. Puisqu'un k_F-échange contient au plus k messages, cela montre que toute composante fortement connexe est de taille au plus k. De plus, si $\mathbf{m} \xrightarrow{\text{RS}} \mathbf{m}'$, alors $\iota(\mathbf{m}) < \iota(\mathbf{m}')$, puisque dans un k_F-échange tous les envois doivent précéder les réceptions. Ainsi, une arête RS ne peut apparaître dans un cycle.

$\Leftarrow$ Soit e une linéarisation de μ. Supposons que le graphe de dépendances étendu $\mathsf{GDE}(e)$ ne contient ni de composante fortement connexe de taille supérieure à k, ni un cycle avec une arête RS. Soit $V_1, \cdots, V_n$ l'ensemble des composantes fortement connexes maximales du graphe de dépendances étendu, listées dans un ordre topologique. Pour un indice i donné, soit $e_i = s_1 \cdots s_m r_1 \cdots r_{m'}$ l'énumération des actions des messages de V_i, définie en prenant d'abord tous les envois de V_i dans l'ordre dans lequel ils apparaissent dans e, puis les réceptions de

V_i dans le même ordre que dans e aussi. L'absence d'arc RS dans les composantes fortement connexes nous assure qu'aucune réception ne doit se trouver avant un envoi. Alors, chaque e_i est mb-réalisable. Posons $e' = e_1 \cdots e_n$, alors $\mathsf{GDE}(e')$ est le même que $\mathsf{GDE}(e)$. Ainsi, e et e' ont le même graphe de dépendances étendu et $msc(e') = msc(e)$. Montrons que e' est mb-réalisable. Par l'absurde, supposons qu'il existe i, j tels qu'il existe un message non couplé dans un canal c_p, $p \in \mathbb{P}$, dans e_i et un message couplé dans c_p dans e_j. Alors, il existe un arc $V_j \overset{\text{SS}}{\dashrightarrow} V_i$, or $V_i \rightarrow^* V_j$ donc, ces deux composantes fortement connexes forment une seule et même composante fortement connexe ce qui est une contradiction. Donc, e' est mb-réalisable. De plus, aussi par hypothèse, $\mid V_i \mid \leq k$ pour tout i, donc chaque e_i est un k-échange et, finalement, μ est k_F-synchrone.

□

Exemple 6.2.2 – La Figure 6.4.a représente le graphe de dépendances étendu du MSC de la Figure 6.3.a. On constate que, comme énoncé dans l'Exemple 6.2.1, les messages doivent appartenir au même k_F-échange car ils appartiennent à la même composante fortement connexe si l'on considère le graphe de dépendances étendu. Celle-ci est de taille 3, le MSC est donc 3_F-synchrone.

La Figure 6.4.b représente le graphe de dépendances étendu du MSC de la Figure 6.3.b. De la même façon, les messages appartiennent à la même composante fortement connexe. Celle-ci contient un arc RS ce qui explique que l'on ne peut pas organiser tous les envois suivis de toutes les réceptions et que donc le MSC correspondant n'est pas k_F-synchrone pour aucun k.

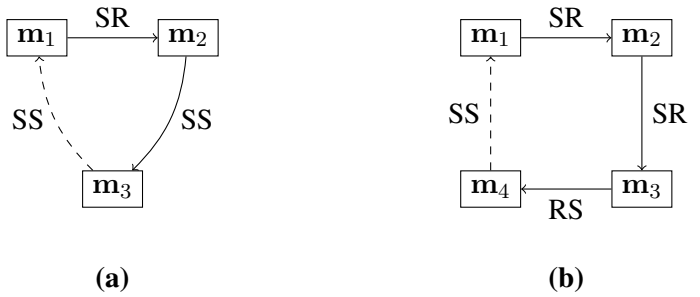


FIGURE 6.4 – Graphe de dépendances étendu du MSC de la Figure 6.3.a en (a) et de la Figure 6.3.b en (b)

6.2.3 La k-synchronisabilité sans ordre

Dans cette section, le but est d'élargir la classe créée par les systèmes k_F-synchronisables tout en conservant la cohérence entre division en k-échanges et exécutions. Pour cela, nous pouvons relâcher la contrainte concernant l'ordre des envois et des réceptions dans un k-échange. Un k-échange définit alors une phase d'actions, sans ordre particulier entre l'ensemble des envois et l'ensemble des réceptions, à la fin de laquelle une porte de synchronisation est passée : on s'assure que tout message non reçu à ce moment sera perdu à jamais. Nous appelerons cette nouvelle classe la *k-synchronisabilité sans ordre*, noté k_{so}-synchronisabilité. Comme pour la k_F-synchronisabilité, nous définisson la k_{so}-synchronisabilité pour les deux types de communication.

Commençons donc par la définition d'un k-échange dans ce nouveau contexte. On parlera alors de k_{so}-échange pour un k-échange n'imposant pas d'ordre entre l'ensemble des envois et celui des réceptions.

Définition 6.2.4 (k_{so}-échange)**.** Un k_{so}-échange est une séquence d'actions $e = a_1 \cdots a_n \in (\mathtt{S} + \mathtt{R})^{\leq 2k}$ telle que $\sharp(e, \mathtt{s}(\_, \_, \_)) \leq k$ et telle que, pour tout $1 \leq i \leq n$, $a_i \in \mathtt{R}$, il existe j tel que $j \in [1..i[$ et $a_j \mapsto a_i$.

On peut adapter la définition d'un MSC k-synchrone à la k_{so}-synchronisabilité, on obtient un MSC k_{so}-synchrone, ainsi qu'une exécution k_{so}-synchronisable.

Définition 6.2.5 (MSC k_{so}-synchrone)**.** Un MSC μ est k_{so}-synchrone en boîte aux lettres, respectivement en pair à pair, s'il existe une linéarisation $e = e_1 \cdots e_n$ de μ telle que :

- pour tout $i \in [1..n]$, e_i est un k_{so}-échange,
- e est une exécution mb-réalisable, respectivement pp-réalisable.

Une exécution e est k_{so}-synchronisable si $msc(e)$ est k_{so}-synchrone.

On note $\text{MSC}_k^{so}(\mathcal{S})$ l'ensemble des MSC k_{so}-synchrones d'un système $\mathcal{S}$. Ainsi, on définit un système k_{so}-synchronisable comme suit.

Définition 6.2.6 (Système k_{so}-synchronisable)**.** Un système $\mathcal{S}$ est k_{so}-synchronisable si $\text{MSC}(\mathcal{S}) = \text{MSC}_k^{so}(\mathcal{S})$.

Poursuivons avec des exemples de k_{so}-échanges.

Exemple 6.2.3 – Le MSC de la Figure 6.5.a est extrait de [Bouajjani et al., 2018a], il n'est pas k-synchrone pour aucun k car tous les messages doivent être dans le même k-échange mais aucune organisation ne permet d'avoir tous les envois suivis de toutes les réceptions. En revanche, il est 4_{so}-synchrone : les 4 messages constituent un 4_{so}-échange.

Le MSC de la Figure 6.5.b est seulement pp-réalisable. De la même façon que le précédent, il est 3_{so}-synchrone mais pas k-synchrone pour aucun k.

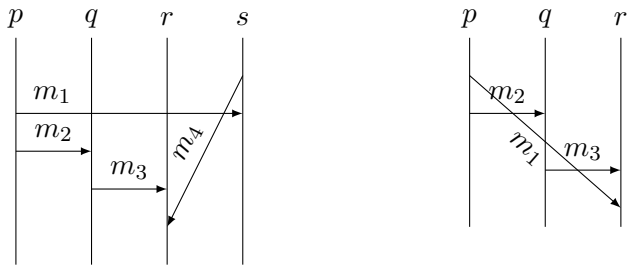


FIGURE 6.5 – Des k_{so}-échanges

Plusieurs constatations peuvent être faites sur cette nouvelle classe. Tout d'abord, et sans difficultés, nous pouvons établir que la classe des systèmes k_F-synchronisables définie précédemment est incluse dans la classe des systèmes k_{so}-synchronisables.

Lemme 6.2.5. *Soit $\mathcal{S}$ un système. Si $\mathcal{S}$ est k_F-synchronisable alors $\mathcal{S}$ est k_{so}-synchronisable.*

Démonstration.

Soit $\mathcal{S}$ un système k_F-synchronisable. Alors, pour tout $\mu \in \text{MSC}(\mathcal{S})$, μ est k_F-synchrone. Par définition, pour tout μ, il existe e com-réalisable, tel que $msc(e) = \mu$ et $e = e_1 \cdots e_n$ avec $e_i \in \mathtt{S}^{\leq k} \cdot \mathtt{R}^{\leq k}$ un k_F-échange. Donc, e_i est également un k_{so}-échange. Donc, pour tout μ, il existe e com-réalisable tel que $msc(e) = \mu$ est e est divisible en k_{so}-échanges. Donc, $\mathcal{S}$ est k_{so}-synchronisable.

□

D'autre part, de façon moins évidente, on peut constater qu'un système k_{so}-synchronisable est également $\exists$-k-com-borné. À la fin de chaque phase, tous les messages sont soit lus, soit non couplés. Les messages non couplés étant ignorés par la définition de la borne existentielle, il ne peut y avoir que k messages couplés mais non lus à la fois, ce qui correspond bien au fait d'être $\exists$-k-com-borné.

Lemme 6.2.6. *Soit $\mathcal{S}$ un système. Si $\mathcal{S}$ est k_{so}-synchronisable alors $\mathcal{S}$ est $\exists$-k-com-borné.*

Démonstration.

Soit $\mathcal{S}$ un système k_{so}-synchronisable. Alors, pour tout $\mu \in \text{MSC}(\mathcal{S})$, il existe e com-réalisable tel que $msc(e) = \mu$ et $e = e_1 \cdots e_n$ où pour tout $1 \leq i \leq n$, e_i est un k_{so}-échange.

Par récurrence sur e, montrons que e est $\exists$-k-bornée.

Base $e = e_1$ donc $\sharp(e, \mathtt{s}(\_,\_,\_)) \leq k$ donc e est k-com-bornée.

Récurrence $e = e_1 \cdots e_{n-1} \cdot e_n$ où $e_1 \cdots e_{n-1}$ est k-com-bornée. Montrons que, pour tout préfixe v de e_n, et tout $p, q \in \mathbb{P}$,

$$min(\sharp(e_1 \cdots e_{n-1} \cdot v, \mathtt{s}(p,q,\_)), \sharp(e_1 \cdots e_n, \mathtt{r}(p,q,\_))) - \sharp(e_1 \cdots e_{n-1} \cdot v, \mathtt{r}(p,q,\_)) \leq k$$

Notons que, $k \geq \sharp(v, \mathtt{s}(p,q,\_)) \geq \sharp(v, \mathtt{r}(p,q,\_))$.
Alors

- soit $\sharp(e_1 \cdots e_{n-1} \cdot v, \mathtt{s}(p,q,\_)) < \sharp(e_1 \cdots e_n, \mathtt{r}(p,q,\_))$ et, les messages envoyés pendant $e_1 \cdots e_{n-1}$ sont soit lus dans leur k_{so}-échange, soit resteront non couplés, alors,

  $$min(\sharp(e_1 \cdots e_{n-1}, \mathtt{s}(p,q,\_)) - \sharp(e_1 \cdots e_{n-1}, \mathtt{r}(p,q,\_))) - \sharp(e_1 \cdots e_{n-1}, \mathtt{r}(p,q,\_)) = 0$$

  Donc
  $$\begin{aligned} &\sharp(e_1 \cdots e_{n-1} \cdot v, \mathtt{s}(p,q,\_)) - \sharp(e_1 \cdots e_{n-1} \cdot v, \mathtt{r}(p,q,\_)) \\ &\leq \sharp(e_1 \cdots e_n, \mathtt{r}(p,q,\_)) - \sharp(e_1 \cdots e_{n-1} \cdot v, \mathtt{r}(p,q,\_)) \\ &= \sharp(e_n, \mathtt{r}(p,q,\_)) - \sharp(e_1 \cdots e_{n-1} \cdot v, \mathtt{r}(p,q,\_)) \\ &\leq \sharp(e_n, \mathtt{r}(p,q,\_)) \leq k \end{aligned}$$

  On peut faire de même avec une communication en boîte aux lettres en remplaçant p par $\_$.

✢ soit $\sharp(e_1 \cdots e_{n-1} \cdot v, \mathtt{s}(p,q,\_)) > \sharp(e_1 \cdots e_n, \mathtt{r}(p,q,\_))$ et

$$\begin{aligned}
&\sharp(e_1 \cdots e_n, \mathtt{r}(p,q,_)) - \sharp(e_1 \cdots e_{n-1} \cdot v, \mathtt{r}(p,q,_)) \\
&= \sharp(e_1 \cdots e_{n-1}, \mathtt{r}(p,q,_)) + \sharp(e_n, \mathtt{r}(p,q,_)) - \sharp(e_1 \cdots e_{n-1}, \mathtt{r}(p,q,_)) - \sharp(v, \mathtt{r}(p,q,_)) \\
&= \sharp(e_n, \mathtt{r}(p,q,_)) - \sharp(v, \mathtt{r}(p,q,_)) \\
&\leq \sharp(e_n, \mathtt{r}(p,q,_)) \leq k
\end{aligned}$$

On peut faire de même avec une communication en boîte aux lettres en remplaçant p par "_".

Ainsi, dans les deux cas, pour tout $p, q \in \mathbb{P}$, et v préfixe de e_n,

$$min(\sharp(e_1 \cdots e_{n-1} \cdot v, \mathtt{s}(p,q,\_)), \sharp(e_1 \cdots e_n, \mathtt{r}(p,q,\_))) - \sharp(e_1 \cdots e_{n-1} \cdot v, \mathtt{r}(p,q,\_)) \leq k$$

Alors, e est k-com-bornée. Donc, tout MSC $\mu \in \text{MSC}(\mathcal{S})$ est $\exists$-k-com-borné. Donc, $\mathcal{S}$ est $\exists$-k-com-borné.

□

6.2.4 Caractérisation graphique de la k_{so}-synchronisabilité

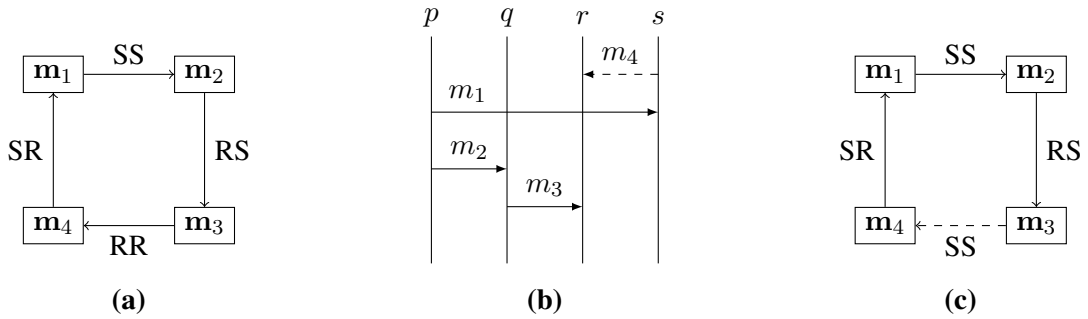
Pour reconnaître une exécution k_{so}-synchronisable, il est possible de caractériser graphiquement un MSC k_{so}-synchrone, tout comme pour la k-synchronisabilité et la k_F-synchronisabilité. Pour être k_{so}-synchrone, la division doit correspondre à une exécution réalisable, ce qui nous pousse à utiliser pour cela le graphe de dépendances étendu. Dans une communication en boîte aux lettres, comme nous l'avons vu avec la k_F-synchronisabilité, le graphe de dépendances étendu nous permet de savoir comment regrouper les messages : les messages d'une même composante fortement connexe appartiennent au même k_{so}-échange. Un système avec une communication en pair à pair nécessite uniquement du graphe de dépendances.

Comme nous n'imposons pas d'ordre entre envois et réceptions dans un k_{so}-échange, on autorise donc la présence d'un arc RS dans une composante fortement connexe, qui indique qu'une réception doit être faite avant l'envoi d'un message dans un même k_{so}-échange. Nous pouvons constater cette caractérisation dans quelques exemples.

Exemple 6.2.4 – Le graphe de dépendances de la Figure 6.6.a est celui du MSC de la Figure 6.5.a. La présence de l'arc RS dans la composante fortement connexe indique que le MSC associé n'est pas k-synchrone pour aucun k, quelque soit le type de communication. Avec la k_{so}-synchronisabilité, aucun ordre entre envois et réceptions ne doit être respecté et l'arc RS ne présente alors plus un problème pour être 4_{so}-synchrone.

Le MSC de la Figure 6.6.b est 1-synchrone. Son graphe de dépendances étendu, Figure 6.6.c, nous montre que tous les messages appartiennent à la même composante fortement connexe et doivent donc appartenir au même 4_{so}-échange. Cette même composante fortement connexe contient un arc RS et donc ce MSC n'est pas k_F-synchrone pour aucun k.

Nous pouvons alors déduire des constatations faites les caractérisations graphiques établies dans le théorème suivant.



(a) **(b)** **(c)**

FIGURE 6.6 – Exemples de caractérisations graphiques

Théorème 6.2.7. *Soit μ un MSC* mb*-réalisable. μ est k_{so}-synchrone si et seulement si son graphe de dépendances étendu* $\mathsf{GDE}(\mu)$ *ne contient pas de composantes fortement connexes de taille supérieure à k.*

Soit μ un MSC pp*-réalisable. μ est k_{so}-synchrone si et seulement si son graphe de dépendances* $\mathsf{GD}(\mu)$ *ne contient pas de composantes fortement connexes de taille supérieure à k.*

Démonstration.

Soit μ un MSC mb-réalisable.

$\Rightarrow$ Si μ est k_{so}-synchrone, alors $\exists e = e_1 \cdots e_n$, e mb-réalisable, telle que $msc(e) = \mu$ où chaque e_i est un k_{so}-échange. Pour chaque sommet $\mathbf{m}$ du graphe de dépendances étendu $\mathsf{GDE}(e)$ il y a exactement un index $\iota(\mathbf{m}) \in [1..n]$ tel que $\mathbf{m} \subseteq e_{\iota(\mathbf{m})}$. Maintenant, observons que, s'il y a un arc de $\mathbf{m}$ à $\mathbf{m}'$ dans le graphe de dépendances étendu, une des actions de $\mathbf{m}$ doit avoir lieu avant une action de $\mathbf{m}'$ dans e, c'est-à-dire, $\iota(\mathbf{m}) \leq \iota(\mathbf{m}')$. Donc, si $\mathbf{m}, \mathbf{m}'$ sont dans la même composante fortement connexe, alors $\iota(\mathbf{m}) = \iota(\mathbf{m}')$ et ils doivent apparaître dans le même k_{so}-échange. Puisqu'un k-échange contient au plus k messages, cela montre que toute composante fortement connexe est de taille au plus k.

$\Leftarrow$ Soit e une linéarisation mb-réalisable de μ. Supposons que le graphe de dépendances étendu $\mathsf{GDE}(e)$ ne contient pas de composante fortement connexe de taille supérieure à k. Soit $V_1, \cdots, V_n$ l'ensemble des composantes fortement connexes maximales du graphe de dépendances étendu, listées dans un ordre topologique. Pour un indice i donné, soit $e_i = a_1 \cdots a_m$ l'énumération des actions des messages de V_i, définie en respectant leur ordre dans e. Posons $e' = e_1 \cdots e_n$, alors $\mathsf{GDE}(e')$ est le même que $\mathsf{GDE}(e)$ car les actions de deux composantes fortement connexes sont concurrentes alors leur ordre n'impacte pas le graphe de dépendances étendu, et l'ordre de deux actions dans la même composante fortement connexe est le même que dans e. Alors, si $\mathsf{GDE}(e') = \mathsf{GDE}(e)$ et donc $msc(e') = msc(e)$. De plus, aussi par hypothèse, $\mid V_i \mid \leq k$ pour tout i, et donc chaque e_i est un k_{so}-échange et, finalement, μ est k_{so}-synchrone.

La preuve pour la communication en pair à pair peut facilement être adaptée de celle-ci. Elle sera alors analogue à la preuve du Théorème 4.1.2, si l'on ignore la contrainte de l'arc RS ici inutile.

□

6.3 Comparaison

Toutes les définitions étant données, nous pouvons à présent les comparer. Dans cette section portera donc sur l'étude des liens qui unient les différentes classes. On s'intéressera aux systèmes $\exists$-k-com-bornés, $\forall$-k-com-bornés, k-synchronisables, k_F-synchronisables et k_{so}-synchronisables. Selon le type de communication du système, boîte aux lettres ou pair à pair, on obtient deux classifications différentes. Elles sont résumées dans les Figures 6.7 et 6.20. Pour chaque intersection de ces diagrammes, nous présenterons un exemple de système correspondant.

Par abus de langage, nous parlerons de la classe des systèmes k-synchronisables de façon générique, il s'agira de la classe contenant tout système $\mathcal{S}$ tel qu'il existe un k' tel que $\mathcal{S}$ est k'-synchronisable. Ceci sera vrai pour toutes les classes de systèmes dont nous parlerons ici.

On trouve dans la littérature des comparaisons concernant les systèmes $\exists$-k-pp-bornés et k-synchronisables et encore d'autres systèmes non précisément décrits ou non évoqués dans cette thèse, d'une part dans [Lange and Yoshida, 2019] mais également dans [Bollig et al., 2021] auquel nous avons participé.

Avant de commencer cette phase de comparaison, nous pouvons établir deux dernières relations, évidentes, mais non explicitées encore. Tout d'abord, si un système est $\forall$-k-com-borné alors il est immédiatement $\exists$-k-com-borné.

Lemme 6.3.1. *Soit $\mathcal{S}$ un système. Si $\mathcal{S}$ est $\forall$-k-com-borné alors il existe $k' \leq k$ tel que $\mathcal{S}$ est $\exists$-k'-com-borné.*

Démonstration.

Soit $\mathcal{S}$ un système $\forall$-k-com-borné. Alors, pour tout $\mu \in \text{MSC}(\mathcal{S})$, μ est $\forall$-k-com-borné et il existe en particulier une exécution k-com-bornée. Donc, μ est $\exists$-k-com-borné. Ainsi, $\mathcal{S}$ est $\exists$-k-com-borné.

□

D'un autre côté, si un système communique en pair à pair et est k-synchronisable alors il est également k_{so}-synchronisable.

Lemme 6.3.2. *Soit μ un MSC pp-réalisable. Si μ est k-synchrone alors μ est k_{so}-synchrone.*

Démonstration.

Soit μ un MSC du système $\mathcal{S}$. Alors par hypothèse, μ est k-synchronisable. D'après, le Lemme 6.2.2, μ est également k_F-synchronisable. Ainsi, par le Lemme 6.2.1, μ est aussi k_{so}-synchronisable.

□

6.3.1 Comparaison pour la communication en pair à pair

On peut alors comparer les différentes classes de systèmes et nous commencerons par le cas de la communication en pair à pair.

Dans une communication en pair à pair, comme nous l'avons déjà démontré avec le Lemme 6.2.2, tout système k_F-synchronisable est également k-synchronisable. On peut également établir que les systèmes k-synchronisables sont k_{so}-synchronisables dans un contexte en pair à pair. Par le Lemme 6.3.2, k-synchrone en pair à pair est également k_{so}-synchrone.

À partir de toutes les propriétés que nous avons vu et de contre-exemples que nous verrons en détail, nous sommes alors capables de déduire la Figure 6.7. Elle représente l'imbrication des différentes classes étudiées.

Toutes ces classes, k-synchronisable, k_F-synchronisable et k_{so}-synchronisable, sont incluses dans la classe des systèmes $\exists$-k-pp-bornés. Enfin, la classe des systèmes $\forall$-k-pp-bornés est incluse dans la classe des systèmes $\exists$-k-pp-borné, et admet une intersection avec deux les autres classes.

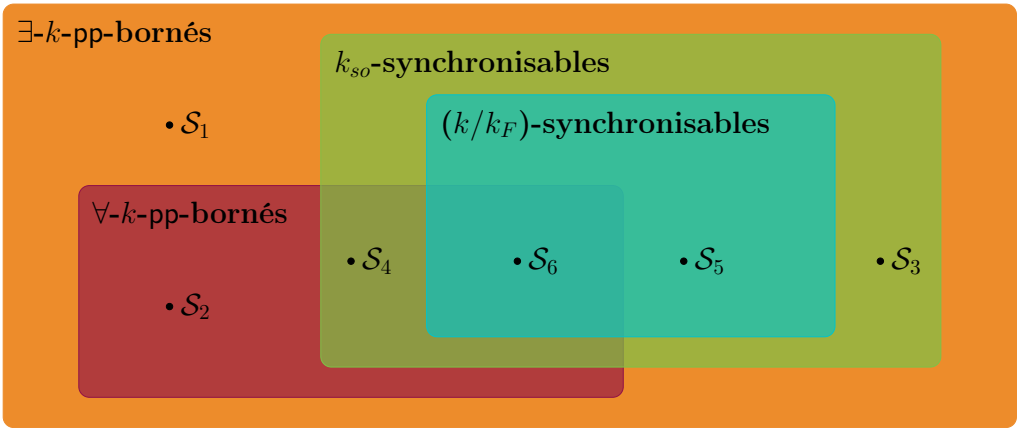


FIGURE 6.7 – Hiérarchie des classes de systèmes communiquant en pair à pair

Comme on peut le constater, les systèmes $\exists$-k-pp-bornés forment une classe englobant toutes les autres. Un système peut d'ailleurs être $\exists$-k-pp-borné sans valider aucune autre propriété.

Exemple 6.3.1 – Un système $\exists$-k-pp-borné C'est le cas du système $\mathcal{S}_1$ représenté en Figure 6.8 : il est uniquement $\exists$-k-pp-borné. On peut observer un comportement possible du système avec le MSC μ_1 représenté en Figure 6.9. On explique le fait que ce système ne soit pas k_{so}-synchronisable par l'entrelacement des messages m_2 et m_3. Toutes les occurrences doivent donc appartenir au même k_{so}-échange, et donc pour tout k, on peut trouver un k_{so}-échange plus grand en ajoutant une itération d'un des deux messages. On est donc capable de construire un k_{so}-échange arbitrairement grand. D'autre part, la répétition du message m_4 rend ce système non $\forall$-k-pp-borné : on peut toujours garder un message m_4 de plus dans le canal $c_{q,r}$ et donc il n'existe pas de k tel que toutes les linéarisations de tous les MSC sont k-pp-bornées.

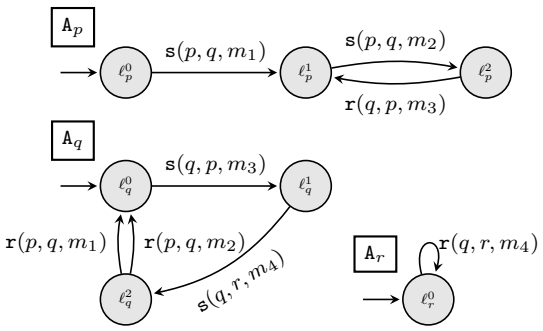


FIGURE 6.8 – Système $\mathcal{S}_1$

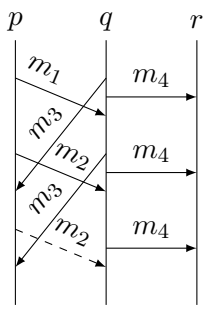


FIGURE 6.9 – MSC μ_1

On reconnaît alors deux motifs présentés ci-dessus, le premier consiste à entrelacer des messages et permet d'obtenir des k_{so}-échanges arbitrairement grands. Le second envoie un message en boucle et empêche d'être $\forall$-k-pp-borné. Si l'on construit un système ne contenant que le premier motif, on obtient alors un système $\exists$-k-pp-borné mais également $\forall$-k-pp-borné mais pas k_{so}-synchronisable.

Exemple 6.3.2 – Un système $\exists$-k-pp-borné et $\forall$-k-pp-borné Le système $\mathcal{S}_2$ représenté en Figure 6.10 est $\exists$-k-pp-borné et $\forall$-k-pp-borné. Il est accompagné d'un comportement possible avec le MSC μ_2 de la Figure 6.11. En effet, comme le message m_3 doit être lu pour pouvoir envoyer la prochaine itération du message m_2 et inversement, toute exécution du système ne peut contenir que, au plus, deux messages dans le même canal, en l'occurrence les messages m_1 et m_2 dans le canal $c_{p,q}$.

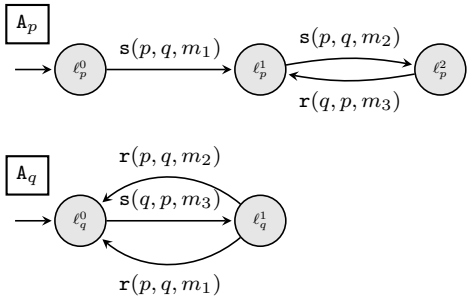


FIGURE 6.10 – Système $\mathcal{S}_2$

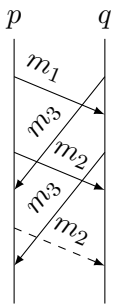


FIGURE 6.11 – MSC μ_2

Regardons à présent les systèmes k_{so}-synchronisables. Dans un premier temps, voyons un système seulement k_{so}-synchronisable (et, forcément, $\exists$-k-pp-borné).

Exemple 6.3.3 – Un système k_{so}-synchronisable C'est le cas du système $\mathcal{S}_3$ représenté en Figure 6.12. Comme on peut le voir sur le MSC $\mu_3 \in \text{MSC}(\mathcal{S}_3)$ de la Figure 6.13. Les messages m_1, m_2, m_3 et m_4 doivent appartenir au même k_{so}-échange. Si l'on construit le graphe de dépendances correspondant, l'on pourrait voir que tous les messages appartiennent à la même composante fortement connexe. Cependant, celle-ci contiendrait également un arc RS entre les messages m_2 et m_3. Ainsi, ce motif nous assure que le système observé n'est pas k-synchronisable. On y retrouve également le message m_5 pouvant être envoyé et reçu en boucle par les processus p et q, ce qui, comme on l'a vu, empêche le système d'être $\forall$-k-pp-borné.

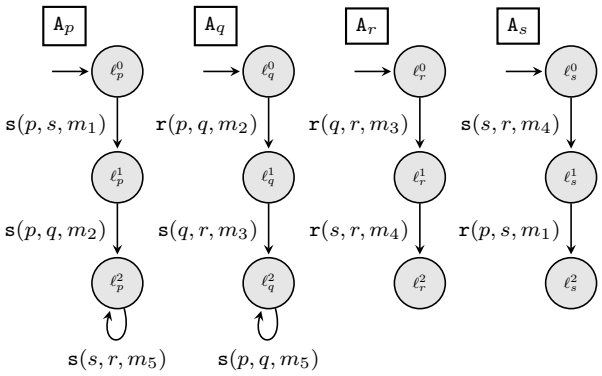


FIGURE 6.12 – Système $\mathcal{S}_3$

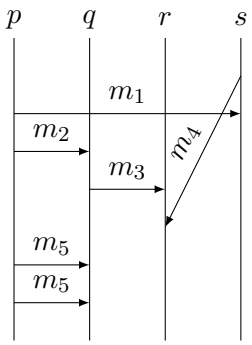


FIGURE 6.13 – MSC μ_3

On observe alors un nouveau motif que forment les messages m_1, m_2, m_3 et m_4 : ils constituent un k_{so}-échange mais pas un k-échange. Encore une fois, si l'on préserve ce motif et qu'on supprime le second, on obtient un système k_{so}-synchronisable et $\forall$-k-pp-borné.

Exemple 6.3.4 – Un système k_{so}-synchronisable et $\forall$-k-pp-borné C'est le cas du système $\mathcal{S}_4$ de la Figure 6.14. On observe un comportement possible avec le MSC μ_4 de la Figure 6.15.

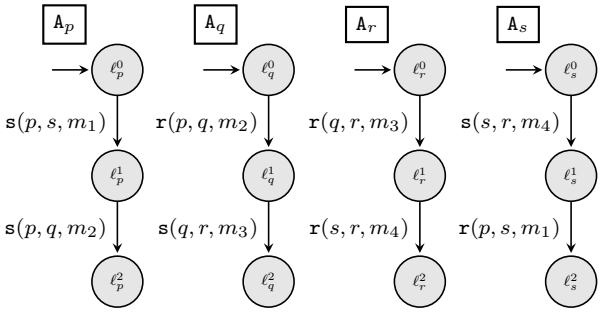


FIGURE 6.14 – Système $\mathcal{S}_4$

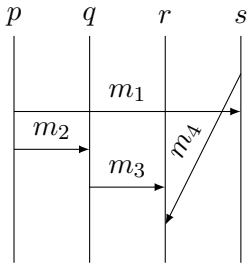


FIGURE 6.15 – MSC μ_4

Restent alors les systèmes k-synchronisables, et donc également k_F-synchronisables.

Exemple 6.3.5 – Un système k-synchronisable On peut alors construire le système $\mathcal{S}_5$ de la Figure 6.16 qui est k-synchronisable, comme on peut facilement le voir avec le MSC μ_5 qui fait partie des comportements possibles. On constate alors également qu'il correspond au schéma empêchant le système d'être $\forall$-k-pp-borné.

FIGURE 6.16 – Système $\mathcal{S}_5$

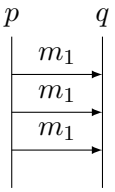


FIGURE 6.17 – MSC μ_5

Enfin, pour permettre au système d'être également $\forall$-k-pp-borné, on peut par exemple alterner envois et réceptions, l'objectif étant d'empêcher les envois successifs et illimités.

Exemple 6.3.6 – Un système k-synchronisable et $\forall$-k-pp-borné C'est ce qu'on retrouve avec le système $\mathcal{S}_6$ de la Figure 6.18, et le MSC représentatif μ_6 de la Figure 6.19.

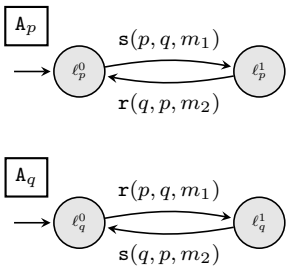


FIGURE 6.18 – Système $\mathcal{S}_6$

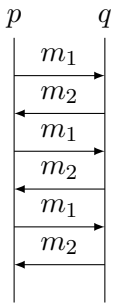


FIGURE 6.19 – MSC μ_6

6.3.2 Comparaison pour la communication en boîte aux lettres

La principale différence avec les systèmes communiquant en boîte aux lettres réside dans le fait que les systèmes k-synchronisables ne soient plus toujours k_F-synchronisables, k_{so}-synchronisables ni même $\exists$-k-mb-bornés. On obtient alors une classe à part, incomparable avec la classe des systèmes $\exists$-k-mb-borné ou $\forall$-k-mb-borné mais également, avec la classe des systèmes k_{so}-synchronisables. Les systèmes k_F-synchronisables sont eux, toujours inclus dans la classe des systèmes k-synchronisables. On peut observer tout cela sur le diagramme de la Figure 6.20.

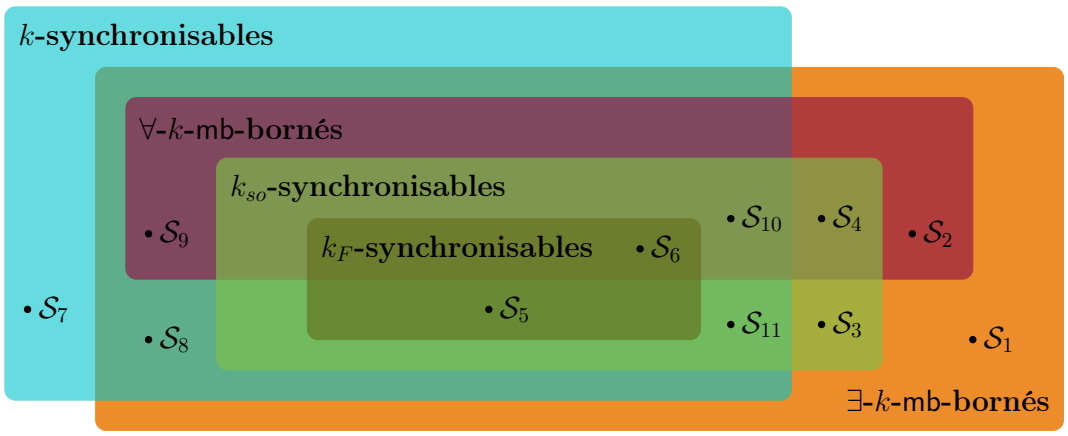


FIGURE 6.20 – Hiérarchie des classes de systèmes communiquant en boîte aux lettres

En effet, en boîte aux lettres, un système peut être k-synchronisable sans être $\exists$-k-mb-borné.

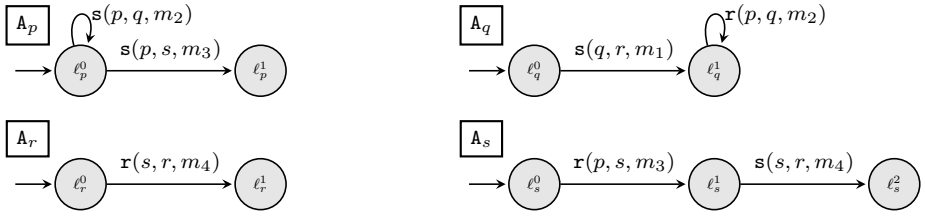


FIGURE 6.21 – Système $\mathcal{S}_7$

Exemple 6.3.7 – Un système k-synchronisable C'est le cas du système $\mathcal{S}_7$ de la Figure 6.21. On observe, dans la Figure 6.22, le MSC $\mu_7 \in \text{MSC}(\mathcal{S}_7)$, et dans la Figure 6.23 le graphe de dépendances étendu correspondant. On peut donc vérifier que μ_7 est bien mb-réalisable, avec l'absence de cycle SS. On constate également, si l'on regarde les arcs du graphe de dépendances seulement (sans l'arc en pointillé qui appartient seulement au graphe de dépendances étendu), que les messages forment chacun un k-échange. Cependant, le graphe de dépendances étendu lui ne contient qu'une seule et même composante fortement connexe contenant tous les messages, y compris toutes les itérations du message m_2. On peut donc construire un k_{so}-échange arbitrairement grand en ajoutant une itération de m_2. Enfin, le message m_2 ne peut être reçu directement après son envoi : sa réception doit être faite après l'envoi du message m_1 et le message m_1 ne peut pas être envoyé avant m_2 auquel cas le message m_4 ne pourra être reçu. Ainsi, toutes les itérations du message m_2 doivent être stockées jusqu'à l'envoi des messages m_4 puis m_1. Alors, pour tout k, on peut trouver un MSC nécessitant un canal c_q plus grand que k en augmentant le nombre d'itérations de m_2.

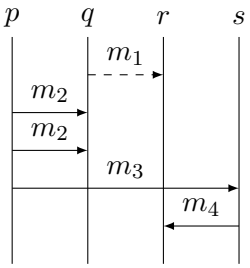


FIGURE 6.22 – MSC μ_7

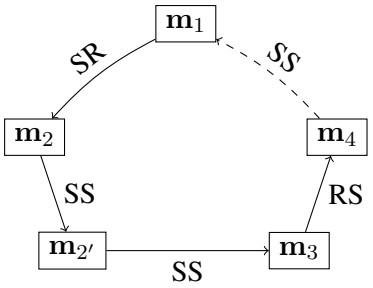


FIGURE 6.23 – GD(μ_7)

Poursuivons avec les systèmes k-synchronisables. Voyons tout d'abord l'intersection avec la classe des systèmes $\exists$-k-mb-bornés.

Exemple 6.3.8 – Un système k-synchronisable et $\exists$-k-pp-borné Le système $\mathcal{S}_8$ de la Figure 6.24 est k-synchronisable et $\exists$-k-pp-borné. Il n'est cependant pas $\forall$-k-pp-borné ni k_{so}-synchronisable à cause de la répétition du message m_2. En effet, les différentes itérations du message m_2 peuvent être toutes faites avant d'être lues, or, comme leur nombre n'est pas borné, ceci implique qu'un nombre non borné de messages peut être stocké dans le canal c_p. D'autre part, pour qu'une linéarisation soit mb-réalisable, l'envoi de m_3 doit se faire avant l'envoi de m_1. Ainsi, tous les messages appartiennent au même k_{so}-échange. Comme dit précédemment, le nombre de messages m_2 n'est pas borné et donc, la taille des k_{so}-échanges produits par ce système ne l'est pas elle aussi, et donc $\mathcal{S}_8$ n'est pas k_{so}-synchronisable pour aucun k.

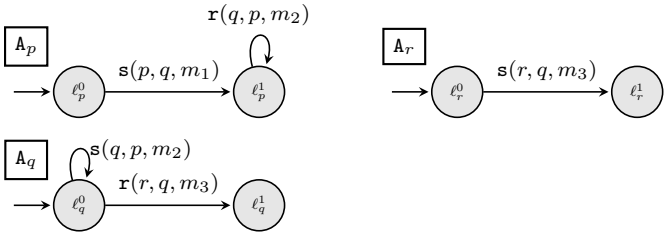


FIGURE 6.24 – Système $\mathcal{S}_8$

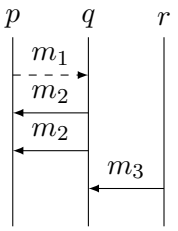


FIGURE 6.25 – MSC μ_8

Pour permettre à un système d'être également $\forall$-k-pp-borné, il faut supprimer le motif de répétition du message et le remplacer tout en laissant la possibilité d'avoir un $_{so}$-échange arbitrairement grand, on peut par exemple remplacer la répétition d'un seul message par deux messages s'alternant.

Exemple 6.3.9 – Un système k-synchronisable et $\forall$-k-pp-borné C'est ce que l'on observe sur le MSC μ_9 de la Figure 6.27 avec les messages m_2 et m_3. Ce MSC représente un comportement possible du système $\mathcal{S}_9$ que l'on peut voir en Figure 6.26. Alterner les messages assure que le précédent doit être lu avant de pouvoir envoyer le suivant, on est alors certains que le nombre de messages dans les canaux est borné. Comme précédemment, tous les messages doivent appartenir au même k_{so}-échange, qui peut donc être arbitrairement grand, et empêche le système d'être k_{so}-synchronisable.

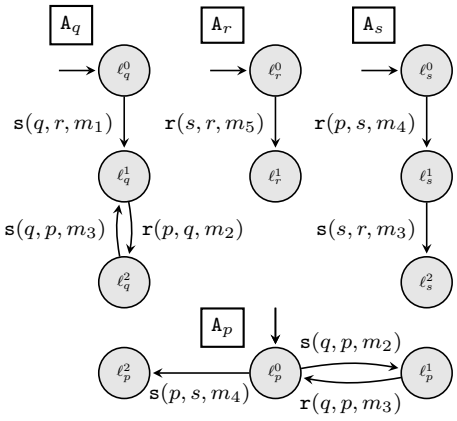


FIGURE 6.26 – Système $\mathcal{S}_9$

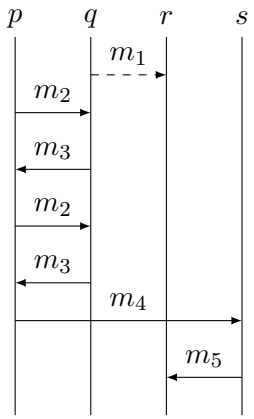


FIGURE 6.27 – MSC μ_9

Pour obtenir un système k_{so}-synchronisable, on doit s'assurer que les tailles des k_{so}-échanges possibles sont bornées.

Exemple 6.3.10 – Un système k-synchronisable et k_{so}-synchronisable C'est le cas dans le système $\mathcal{S}_{10}$ de la Figure 6.28. Le nombre de messages est fini ce qui nous assure d'être $\exists$-k-mb-borné, $\forall$-k-mb-borné et k_{so}-synchronisable. Le système est également k-synchronisable car chaque message est indépendant, comme on peut le voir dans le MSC μ_{10} de la Figure 6.29. Cependant, pour organiser les actions de façon à être réalisable, il faut envoyer le message m_4 avant le message m_1, et donc, tous les messages doivent appartenir au même k-échange. Or la réception du message m_3 étant nécessaire à l'envoi du message m_4, on ne peut pas obtenir un k_F-échange.

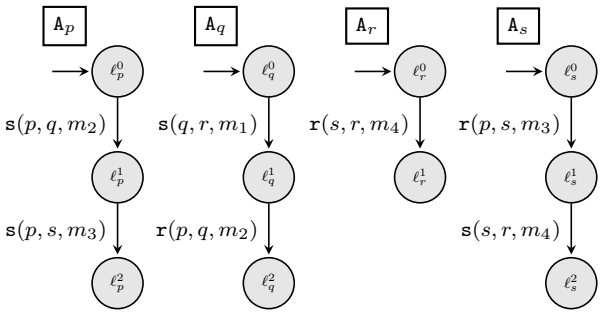


FIGURE 6.28 – Système $\mathcal{S}_{10}$

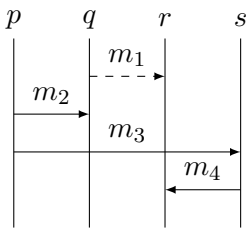


FIGURE 6.29 – MSC μ_{10}

Pour terminer, dans la communication en boîte aux lettres, on peut trouver des systèmes $\exists$-k-mb-bornés, k-synchronisables et k_{so}-synchronisables, qui ne sont pas k_F-synchronisables ni $\forall$-k-mb-bornés.

Exemple 6.3.11 – On retrouve alors le système $\mathcal{S}_{11}$ de la Figure 6.30. En effet, comme nous pouvons le voir sur le MSC μ_{11} de la Figure 6.31, le schéma vu précédemment crée un k_{so}-échange contenant les messages m_1, m_2, m_3 et m_4 empêchant le système d'être k_F-synchronisable. L'ajout du message m_5 répété empêche, lui, d'être $\forall$-k-mb-borné.

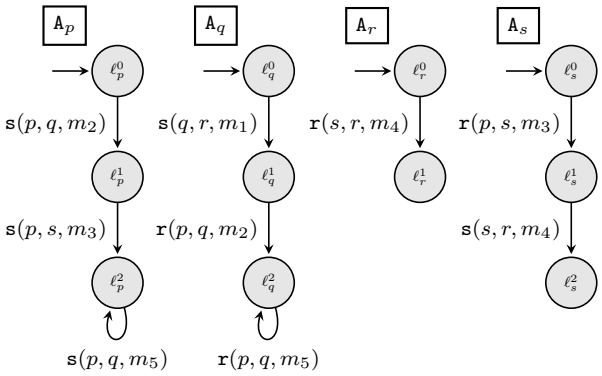


FIGURE 6.30 – Système $\mathcal{S}_{11}$

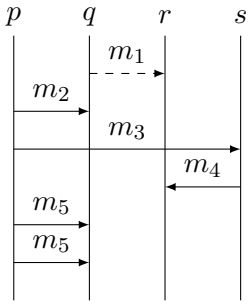


FIGURE 6.31 – MSC μ_{11}

Les dernières intersections peuvent être représentées par les mêmes systèmes que pour la communication en pair à pair.

Exemple 6.3.12 – Les autres intersections Le système $\mathcal{S}_6$ de la Figure 6.18, déjà vu précédemment, correspond à un système $\exists$-k-mb-borné, $\forall$-k-mb-borné, k-synchronisable, k_{so}-synchronisable et k_F-synchronisable, comme c'était déjà le cas dans la communication en pair à pair.

De la même façon, le système $\mathcal{S}_5$ de la Figure 6.16, reste un bon exemple de système $\exists$-k-mb-borné, k-synchronisable, k_{so}-synchronisable et k_F-synchronisable, le système $\mathcal{S}_3$ de la Figure 6.12 est $\exists$-k-mb-borné et k_{so}-synchronisable, le système $\mathcal{S}_2$ est $\exists$-k-mb-borné et $\forall$-k-mb-borné et enfin le système $\mathcal{S}_1$ est uniquement $\exists$-k-mb-borné.

6.4 Preuves des problèmes de d'accessibilité et d'appartenance

Les systèmes k_F-synchronisables et k_{so}-synchronisables sont tous $\exists$-k-pp-bornés. Le problème de l'accessibilité dans un système $\exists$-k-pp-borné est décidable. Ainsi, il est évident que ce problème est décidable pour les systèmes k_F-synchronisables et k_{so}-synchronisables. Cependant, par souci de complétude, nous apportons dans cette section les preuves de la décidabilité de l'accessibilité d'un état de contrôle pour de tels systèmes.

D'autre part, les travaux effectués dans [Bollig et al., 2021] prouvent que pour le type de systèmes communicants étudiés, l'appartenance à des classes comme $\exists$-k-mb-borné ou k_F-synchronisable est décidable, et élargit ce résultat à bien d'autres classes. Ainsi, savoir si un système quelconque est k_F-synchronisable ou k_{so}-synchronisable est décidable d'après [Bollig et al., 2021]. Cependant, et encore une fois par souci de complétude, nous prouverons que, pour un k donné, savoir si le système étudié est k_F-synchronisable, et respectivement k_{so}-synchronisable, est décidable.

6.4.1 La décidabilité de l'accessibilité dans un système k_F-synchronisable

Pour établir la décidabilité du problème de l'accessibilité dans un système k_F-synchronisable, nous utiliserons les résultats du Lemme 6.2.1 : tout MSC k_F-synchrone est également k'-synchrone, pour un k' plus petit ou égal à k. Ainsi, les systèmes k_F-synchronisables sont une sous-classe des systèmes k-synchronisables, et l'accessibilité d'un état de contrôle étant décidable dans un système k-synchronisable, elle l'est également dans un système k_F-synchronisable.

Théorème 6.4.1. *Soit $\mathcal{S}$ un système k_F-synchronisable, savoir si un état de contrôle est accessible dans $\mathcal{S}$ est décidable.*

Démonstration.

Soit $\mathcal{S}$ est un système k_F-synchronisable. On sait par le Lemme 6.2.1 que tout MSC $\mu \in \text{MSC}(\mathcal{S})$ est k-synchrone. Donc, $\mathcal{S}$ est k-synchrone. D'après le Théorème 4.2.2, l'accessibilité d'un état de contrôle est décidable dans $\mathcal{S}$.

□

6.4.2 Décidabilité de la k_F-synchronisabilité - Sans paramètre k

Dans cette section, nous montrerons qu'il est décidable de déterminer s'il existe, pour un système $\mathcal{S}$ quelconque, un k tel que $\mathcal{S}$ k_F-synchronisable.

Par définition, on sait qu'un système ne peut pas être k_F-synchronisable sans être k-synchronisable. On sait également qu'il est décidable de savoir s'il existe un k tel qu'un système donné est k-synchronisable. Alors, on peut en premier lieu tester notre système à la k-synchronisabilité. S'il n'existe pas de k tel que notre système est k-synchronisable, alors notre système n'est pas non plus k_F-synchronisable, pour aucun k. Si notre système est k-synchronisable, nous pouvons alors nous interroger sur la k_F-synchronisabilité de notre système. Si la communication est en pair à pair, alors, nous savons par le Lemme 6.2.2 que le système observé est k_F-synchronisable. Toute cette section ne concerne alors que les systèmes communiquant en boîte aux lettres.

Nous pouvons alors rappeler la différence entre la k-synchronisabilité et la k_F-synchronisabilité. Pour qu'un système k-synchronisable soit k'_F-synchronisable, il faut que chaque MSC du système admette une exécution réalisable divisible en k'_F-échanges.

Si un système est k-synchronisable, on peut étudier l'ensemble des comportements possibles grâce à la transition $\xRightarrow[\mathsf{real}]{e,k}$ définissant les k-échanges et permettant de les concaténer et d'obtenir les MSC du système.

Grâce au langage décrit par $\xRightarrow[\mathsf{real}]{e,k}$, on peut alors répondre à la question de la présence de messages non couplés pouvant être répétés dans une exécution. Ces messages seraient alors reconnaissables grâce à la présence d'un un k-échange contenant un message non couplé et appartenant à un cycle dans le système de transition ainsi défini.

Il faudra également s'assurer que les MSC du système peuvent être divisés en k_F-échange, c'est-à-dire que la division en k_F-échanges corresponde à une exécution mb-réalisable. Nous verrons en temps voulu les détails de cette étape qui s'appuie sur une étude du langage décrit par $\xRightarrow[\mathsf{real}]{e,k}$.

Nous notons $\exp^k_{\mathsf{real}}(\mathcal{S})$ l'expression régulière qui décrit son langage et utiliserons sa Forme Normale Disjonctive (DNF). Rappelons que pour toute expression régulière, il existe une expression équivalente composée de sous-expression reliée par des opérateurs de choix (voir [Cousot, 2019, Chapitre 5] pour plus de détails). Plus précisément, nous décomposerons ce langage grâce à sa DNF pour obtenir un ensemble de sous-expressions régulières sans opérateurs de choix que nous appellerons *chemin*.

On définit cet ensemble d'expressions régulières $\textsc{Dnf}^k_{\mathsf{real}}(\mathcal{S})$ pour un système $\mathcal{S}$ comme suit. On note $\mathcal{L}(r)$ le langage représenté par une expression régulière r.

Définition 6.4.1 (Ensemble $\textsc{Dnf}^k_{\mathsf{real}}(\mathcal{S})$)**.** Soit $\exp^k_{\mathsf{real}}(\mathcal{S})$ l'expression régulière décrivant le langage du système de transition $\xRightarrow[\mathsf{real}]{e,k}$. $\textsc{Dnf}^k_{\mathsf{real}}(\mathcal{S})$ est l'ensemble fini des expressions régulières tel que :

- $w \in \mathcal{L}(\exp^k_{\mathsf{real}}(\mathcal{S}))$ si et seulement si $\exists d \in \textsc{Dnf}^k_{\mathsf{real}}(\mathcal{S}), w \in \mathcal{L}(d)$
- et, pour tout $d \in \textsc{Dnf}^k_{\mathsf{real}}(\mathcal{S})$, d ne contient pas d'opérateur de choix.

Nous avons en notre possession d'un ensemble d'expressions régulières représentant les MSC du système et nous souhaitons savoir s'ils peuvent être divisés en k_F-échanges en suivant une exécution mb-réalisable. Nous savons également que le graphe de dépendances étendu et ses composantes fortement connexes peuvent nous donner une division respectant ces contraintes.

La stratégie est alors la suivante : construire le graphe de dépendances étendu de chacun des chemins afin de vérifier

- l'absence d'arc RS dans toute composante fortement connexe,
- l'existence d'une borne sur la taille des k_F-échanges.

Ce second point peut ne pas être vérifié si un message peut être répété et que toutes les itérations doivent appartenir au même k_F-échange. Ainsi, il faut un traitement particulier des opérateurs de Kleene dans les chemins pour que ces cas soient repérés. Cela permettra également de découvrir des arcs RS qui ne pouvaient apparaître qu'en répétant un message.

Ce traitement consiste à dédoubler tout message pouvant être répété : on considérera alors la première et la dernière itération dues à une étoile. Commençons donc par la modification des chemins.

Définition 6.4.2 (Chemin avec duplication)**.** Soit d un chemin. On définit $d^{\circledast}$ tel que :

$$d^{\circledast} = emb(dup(d))$$

où la fonction $dup(\cdot)$ duplique les actions :

$$dup(d) = \begin{cases} dup(d_1) \cdot dup(d_2) & \text{si } d = d_1 \cdot d_2 \\ dup(d_1) \cdot dup(d_1) & \text{si } d = (d_1)^* \end{cases}$$

et la fonction $emb(\cdot)$ ajoute en exposant un indice supérieur au précédent à chaque message en commençant par 0 pour distinguer les différentes occurrences d'un même $a \in \mathbb{A}$.

Exemple 6.4.1 – Si on considère le système qui permet d'obtenir le MSC de la Figure 6.32, il existe un chemin $d \in \text{DNF}^k_{\mathsf{real}}(\mathcal{S})$ tel que

$$d = \mathtt{s}(p,q,m_1)(\mathtt{s}(q,p,m_2)\mathtt{r}(q,p,m_2))^*\mathtt{s}(r,q,m_3)\mathtt{r}(r,q,m_3)$$

Donc nous avons

$$d^{\circledast} = \mathtt{s}(p,q,m_1^{\mathbf{0}}) \cdot \mathtt{s}(q,p,m_2^{\mathbf{0}}) \cdot \mathtt{r}(q,p,m_2^{\mathbf{0}}) \cdot \mathtt{s}(q,p,m_2^{\mathbf{1}}) \cdot \mathtt{r}(q,p,m_2^{\mathbf{1}}) \cdot \mathtt{s}(r,q,m_3^{\mathbf{0}}) \cdot \mathtt{r}(r,q,m_3^{\mathbf{0}})$$

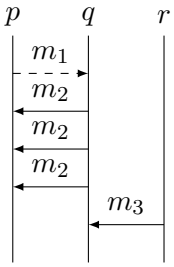


FIGURE 6.32 – Un MSC avec répétition d'un message

Nous pouvons alors construire le graphe de dépendances étendu d'un tel chemin avec duplication.

Définition 6.4.3 (Graphe de dépendances étendu d'un chemin $d^{\circledast}$)**.** Soit d un chemin. Soit $\mathsf{GDE}(d^{\circledast}) = (E, V)$ le graphe de dépendances étendu correspondant défini comme suit :

$$E = \{[\![d^{\circledast}\lfloor_p]\!]_E \mid p \in \mathbb{P}\} \cup E' \qquad V = \{[\![d^{\circledast}\lfloor_p]\!]_V \mid p \in \mathbb{P}\}$$

✣ où $d^{\circledast}\lfloor_p$ est la projection de $d^{\circledast}$ sur le processus $p \in \mathbb{P}$ telle que :

$$d^{\circledast}\lfloor_p = \begin{cases} \mathtt{s}(p,q,m) & \text{si } d^{\circledast} = \mathtt{s}(p,q,m) \text{ pour } q \in \mathbb{P}, m \in \mathbb{V} \\ \mathtt{r}(q,p,m) & \text{si } d^{\circledast} = \mathtt{r}(q,p,m) \text{ pour } q \in \mathbb{P}, m \in \mathbb{V} \\ d_1\lfloor_p \cdot d_2\lfloor_p & \text{si } d^{\circledast} = d_1 \cdot d_2 \\ \varepsilon & \text{sinon} \end{cases}$$

- où $[\![d^{\circledast}\lfloor_p]\!] = ([\![d^{\circledast}\lfloor_p]\!]_V, [\![d^{\circledast}\lfloor_p]\!]_E, [\![d^{\circledast}\lfloor_p]\!]_F)$ est son encodage avec $[\![d^{\circledast}\lfloor_p]\!]_V$ l'ensemble des sommets, $[\![d^{\circledast}\lfloor_p]\!]_E$ l'ensemble des arcs et $[\![d^{\circledast}\lfloor_p]\!]_F$ la dernière action liée telle que :

$$[\![d^{\circledast}\lfloor_p]\!] = \begin{cases} (\{\mathbf{m}\}, \emptyset, a) & \text{si } d^{\circledast}\lfloor_p = a, \\ ([\![d_1]\!]_V \cup \{m\}, [\![d_1]\!]_E \cup \{m' \xrightarrow{\text{XY}} m\}, a) & \text{si } d^{\circledast}\lfloor_p = d_1 \cdot a \end{cases}$$

pour $a = x(q, r, m)$, $[\![d_1]\!]_F = y(q', r', m')$,

avec $X = \begin{cases} S \text{ si } a \in \mathtt{S} \\ R \text{ si } a \in \mathtt{R} \end{cases}$ et $Y = \begin{cases} S \text{ si } [\![d_1]\!]_F \in \mathtt{S} \\ R \text{ si } [\![d_1]\!]_F \in \mathtt{R} \end{cases}$

- et où E' est l'ensemble des arcs déduits des RÈGLES 1 à 5 (page 53).

Exemple 6.4.2 – Poursuivons avec le chemin avec duplication obtenu dans l'Exemple 6.4.1. Nous avons

$$d^{\circledast} = \mathtt{s}(p, q, m_1^{\mathbf{0}}) \cdot \mathtt{s}(q, p, m_2^{\mathbf{0}}) \cdot \mathtt{r}(q, p, m_2^{\mathbf{0}}) \cdot \mathtt{s}(q, p, m_2^{\mathbf{1}}) \cdot \mathtt{r}(q, p, m_2^{\mathbf{1}}) \cdot \mathtt{s}(r, q, m_3^{\mathbf{0}}) \cdot \mathtt{r}(r, q, m_3^{\mathbf{0}})$$

On obtient pour chaque processus :

$$\begin{array}{ll} d^{\circledast}\lfloor_p = \mathtt{s}(p, q, m_1^{\mathbf{0}}) \cdot \mathtt{r}(q, p, m_2^{\mathbf{0}}) \cdot \mathtt{r}(q, p, m_2^{\mathbf{1}}) & [\![d^{\circledast}\lfloor_p]\!] = (\{\mathbf{m}_1^{\mathbf{0}}, \mathbf{m}_2^{\mathbf{0}}, \mathbf{m}_2^{\mathbf{1}}\}, \\ & \quad \{\mathbf{m}_1^{\mathbf{0}} \xrightarrow{SR} \mathbf{m}_2^{\mathbf{0}}, \mathbf{m}_2^{\mathbf{0}} \xrightarrow{RR} \mathbf{m}_2^{\mathbf{1}}\}, \mathbf{m}_2^{\mathbf{1}}) \\ d^{\circledast}\lfloor_q = \mathtt{s}(q, p, m_2^{\mathbf{0}}) \cdot \mathtt{s}(q, p, m_2^{\mathbf{1}}) \cdot \mathtt{r}(r, q, m_3^{\mathbf{0}}) & [\![d^{\circledast}\lfloor_q]\!] = (\{\mathbf{m}_2^{\mathbf{0}}, \mathbf{m}_2^{\mathbf{1}}, \mathbf{m}_3^{\mathbf{0}}\}, \\ & \quad \{\mathbf{m}_2^{\mathbf{0}} \xrightarrow{SS} \mathbf{m}_2^{\mathbf{1}}, \mathbf{m}_2^{\mathbf{1}} \xrightarrow{SR} \mathbf{m}_3^{\mathbf{0}}\}, \mathtt{r}(r, q, m_3^{\mathbf{0}}) \\ d^{\circledast}\lfloor_r = \mathtt{s}(r, q, m_3^{\mathbf{0}}) & [\![d^{\circledast}\lfloor_r]\!] = (\{\mathbf{m}_3^{\mathbf{0}}\}, \emptyset, \mathtt{s}(r, q, m_3^{\mathbf{0}})) \end{array}$$

Alors, le graphe de dépendances étendu auquel nous avons ajouté les arcs définis par les RÈGLES 1 à 5 est représenté en Figure 6.33. On observe une composante fortement connexe contenant la première et la dernière itération du même message, le message m_2, donc ce message peut être répété autant de fois que l'on souhaite et toutes les itérations appartiendront au même k_F-échange. Nous pouvons donc construire un k_F-échange arbitrairement grand, ce qui empêche le système d'être k_F-synchronisable, quelque soit le k.

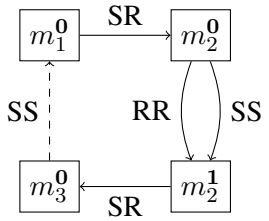


FIGURE 6.33 – Graphe de dépendances étendu du MSC de la Figure 6.32

Une fois le graphe de dépendances étendu construit, on peut vérifier que chaque composante fortement connexe représente un k_F-échange valide, de taille bornée et sans arc RS. On note $\text{COMP}^k_{\mathsf{real}}(\mathcal{S})$ l'ensemble de toutes les composantes fortement connexes de tous les graphes de

dépendances étendus pour un système et un k : Soit

$$\textsc{Comp}^k_{\mathsf{real}}(\mathcal{S}) = \{(E', V') \mid (E', V') \text{ est une composante fortement connexe,} \\ E' \subseteq E, V' \subseteq V, \mathsf{GDE}(d^{\circledast}) = (E, V), d \in \textsc{Dnf}^k_{\mathsf{real}}(\mathcal{S})\}$$

Si les conditions sont respectées, on peut déduire que le système est k'_F-synchronisable, avec $k' = max(k, \mathrm{M}_{comp})$ où M_{comp} est la taille de la plus grande composante fortement connexe dans $\textsc{Comp}^k_{\mathsf{real}}(\mathcal{S})$.

Lemme 6.4.2. *Soit $\mathcal{S}$ un système où* $\mathsf{com} = \mathsf{mb}$*, s'il existe $(E, V) \in \textsc{Comp}^k_{\mathsf{real}}(\mathcal{S})$, $\mathbf{m}$ et $i, j \in \mathbb{N}$ tels que $\mathbf{m^i}, \mathbf{m^j} \in V$ alors il n'existe pas de k tel que $\mathcal{S}$ est k_F-synchronisable.*

Démonstration.

Soit $\mathcal{S}$ un système et $d \in \textsc{Dnf}^k_{\mathsf{real}}(\mathcal{S})$ tel que $\mathsf{GDE}(d^{\circledast})$ contient une composante fortement connexe avec deux itérations du même message. Alors, on a $(E, V) \subseteq \mathsf{GDE}(d^{\circledast})$ tel que $\mathbf{m^x}, \mathbf{m^y} \in V$, $m \in \mathbb{V}, x, y \in \mathbb{N}, x < y$ and $\mathbf{m^x} \dashrightarrow^* \mathbf{m^y} \dashrightarrow^* \mathbf{m^x} \in E$.

Comme $x < y$, il est évident que $\mathbf{m^x} \xrightarrow{\text{SS,RR}} \mathbf{m^y}$. Ainsi, on ne peut pas avoir $\mathbf{m^y} \rightarrow^* \mathbf{m^x}$. Alors, le chemin depuis $\mathbf{m^y}$ vers $\mathbf{m^x}$ contient un arc en pointillé et seulement la RÈGLE 5 permet de créer un tel chemin entre deux messages :

$$\mathbf{m^y} \dashrightarrow^* \mathbf{m} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_1 \dashrightarrow^* \mathbf{m^x}$$

avec $\mathbf{m}, \mathbf{m}_1 \in V$ tels que $\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \mathsf{proc}_{\mathtt{R}}(\mathbf{m}_1)$, $\mathbf{m}$ couplé et $\mathbf{m}_1$ non couplé.

Maintenant, soit e une exécution contenant au moins une itération de plus de m tel que $msc(e) = msc(e')$ et $e' \in \mathcal{L}(d)$. Soit $m^{\mathbf{z}}$ une de ces itérations. Alors,

$$\mathbf{m^y} \dashrightarrow^* \mathbf{m} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_1 \dashrightarrow^* \mathbf{m^x}$$

appartient à $\mathsf{GDE}(msc(e))$.

- Supposons que $z < x < y$, alors $\mathbf{m^z} \xrightarrow{\text{SS,RR}} \mathbf{m^x} \xrightarrow{\text{SS,RR}} \mathbf{m^y} \dashrightarrow^* \mathbf{m} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_1 \dashrightarrow^* \mathbf{m^x}$.
  Pour $a_i \in \mathbf{m}_1 \cap \mathtt{S}$ et $e_j \in \mathbf{m^z} \cap \mathtt{S}$,
  - soit $i < j$ et donc, comme $\mathbf{m}_1 \dashrightarrow^* \mathbf{m^x}$, on a aussi $\mathbf{m}_1 \dashrightarrow^* \mathbf{m^z}$;
  - ou $j < i$ et donc l'envoi de $\mathbf{m}_1$ est entre ceux de $\mathbf{m^z}$ et de $\mathbf{m^x}$ et peut être répété lui aussi, et on a donc, avec une autre répétition de $\mathbf{m}_1$ notée $\mathbf{m}_1^{\mathbf{x'}}$: $\mathbf{m} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_1^{\mathbf{x'}} \dashrightarrow^* \mathbf{m^z}$
- Supposons que $x < z < y$. Alors, immédiatement :

$$\mathbf{m^z} \xrightarrow{\text{SS,RR}} \mathbf{m^y} \rightarrow^* \mathbf{m} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_1 \rightarrow^* \mathbf{m^x} \xrightarrow{\text{SS,RR}} \mathbf{m^z}$$

- Supposons que $x < y < z$. Pour $a_i \in \mathbf{m} \cap \mathtt{S}$ et $a_j \in \mathtt{S}$
  - soit $j < i$ et donc, comme $\mathbf{m^y} \dashrightarrow^* \mathbf{m}$, on a $\mathbf{m^z} \dashrightarrow^*$,

- ✻ soit $i < j$ et donc l'envoi de $\mathbf{m}$ est entre ceux de $\mathbf{m^y}$ et de $\mathbf{m^z}$ et il peut être répété également et donc on aura, pour une autre répétition de $\mathbf{m}$ notée $\mathbf{m^{x'}}$: $\mathbf{m^z} \dashrightarrow^* \mathbf{m^{x'}} \overset{\text{SS}}{\dashrightarrow} \mathbf{m}_1$.

Enfin, peu importe la place de l'autre itération $\mathbf{m^z}$, $\mathbf{m^z}$ appartient à la composante fortement connexe, c'est-à-dire, $\mathbf{m^z} \in V$. Alors, comme il existe un nombre non borné de répétitions de $\mathbf{m}$ et toutes ces répétitions appartiendront à la même composante fortement connexe dans le graphe de dépendances étendu, il n'existe pas de borne sur la taille de la composante fortement connexe. Comme tous les messages sont dans la même composante fortement connexe, ils doivent appartenir au même k-échange (Théorème 6.2.4), alors, pour tout $k' > k$, on peut construire un k'-échange et on ne peut donc pas être k_F-synchronisable pour aucun k.

□

Lemme 6.4.3. *Soit $\mathcal{S}$ un système où* $\mathsf{com} = \mathsf{mb}$. *S'il existe $(E, V) \in \text{COMP}^k_{\mathsf{real}}(\mathcal{S})$, $\mathbf{m}, \mathbf{m}'$ tel que $\mathbf{m} \xrightarrow{\text{RS}} \mathbf{m}' \dashrightarrow^* \mathbf{m}$ alors il n'existe pas de k tel que $\mathcal{S}$ est k_F-synchronisable.*

Démonstration.

Soit $\mathcal{S}$ un système et $d \in \text{DNF}^k_{\mathsf{real}}(\mathcal{S})$ tel que $\mathsf{GDE}(d^{\circledast})$ contient une composante fortement connexe (E, V) avec un arc RS. Alors, on a $\mathbf{m} \xrightarrow{\text{RS}} \mathbf{m}' \dashrightarrow \mathbf{m}$, pour $\mathbf{m}, \mathbf{m}' \in V$. On peut construire une exécution $e \in \mathcal{L}(d)$ telle que $\mathbf{m} \xrightarrow{\text{RS}} \mathbf{m}' \dashrightarrow \mathbf{m}$ apparaît dans $\mathsf{GDE}(e)$. En effet, si d contient une ou plusieurs étoiles de Kleene, prenons e avec exactement deux itérations de chaque et donc $\mathsf{GDE}(d^{\circledast}) = \mathsf{GDE}(e)$. Alors, e est une exécution avec un graphe de dépendances contenant une composante fortement connexe avec un arc RS. D'après le Théorème 6.2.4, e ne peut être k_F-synchronisable pour aucun k et donc, $\mathcal{S}$ n'est pas k_F-synchronisable pour aucun k.

□

Enfin, nous sommes alors capables de savoir s'il existe un k tel qu'un système donné communiquant en boîte aux lettres est k_F-synchronisable. *

Théorème 6.4.4. *Soit $\mathcal{S}$ un système k-synchronisable où* $\mathsf{com} = \mathsf{mb}$. *Savoir s'il existe un k tel que $\mathcal{S}$ est k_F-synchronisable est décidable.*

Démonstration.

Considérons les deux affirmations suivantes :

A. Il existe $comp \in \text{COMP}^k_{\mathsf{real}}(\mathcal{S})$ tel que $comp$ contient deux itérations d'un même message.

B. Il existe $comp \in \text{COMP}^k_{\mathsf{real}}(\mathcal{S})$ tel que $comp$ contient un arc RS.

Soit $\mathcal{S}$ un système k-synchronisable.

$\Rightarrow$ Maintenant, supposons que A soit vraie : d'après le Lemme 6.4.2, $\mathcal{S}$ n'est pas k_F-synchronisable pour aucun k.

∗. Le travail effectué dans cette section apporte une preuve alternative à celle présentée pour les systèmes k-synchronisables. Cependant, celle-ci ne pourrait être appliquée aux systèmes k-synchronisables car elle ne serait pas complète. Des cas problématiques seraient ici manqués par cette démarche.

Enfin, supposons que B soit vraie : alors, d'après le Lemme 6.4.3, $\mathcal{S}$ n'est pas k_F-synchronisable pour aucun k.

$\Leftarrow$ Supposons que A et B soient fausses.

Montrons que chaque exécution du système peut être divisée en k'-échanges.

Par contradiction, supposons qu'il existe $e \in \text{EX}(\mathcal{S})$ tel que e n'est pas k'_F-synchronisable.

- Soit e n'est pas k_F-synchronisable pour aucun k, ce qui implique que $\mathsf{GDE}(e)$ contient une composante fortement connexe avec un arc RS. On sait que toute exécution de $\mathcal{S}$ correspond à un MSC représenté dans $\mathcal{L}(\exp^k_{\mathsf{real}}(\mathcal{S}))$. Donc, il existe une séquence e' de k-échanges dans $\mathcal{L}(\exp^k_{\mathsf{real}}(\mathcal{S}))$ tel que $msc(e') = msc(e)$, $msc(e')$ est k-synchrone, alors il n'existe pas d'arc RS dans aucune composante fortement connexe du graphe de dépendances. Comme $\mathsf{GDE}(e)$ contient une composante fortement connexe avec un arc RS, il contient aussi un arc $\overset{\text{SS}}{\dashrightarrow}$ construit par la RÈGLE 5 (les autres règles réduisent les chemins, mais n'en créent pas). Ceci peut arriver seulement si le système est dans une communication en boîte aux lettres. Sinon, si le système est en pair à pair, il existe une contradiction. Donc, supposons que le système communique en boîte aux lettres. On peut déduire qu'il existe un message non couplé m et un message couplé m', envoyés au même destinataire, tels que la RÈGLE 5 ajoute un arc $\mathbf{m}' \overset{\text{SS}}{\dashrightarrow} \mathbf{m}$.
  Par définition de $\text{DNF}^k_{\mathsf{real}}(\mathcal{S})$, il existe $d \in \text{DNF}^k_{\mathsf{real}}(\mathcal{S})$ tel que $e \in \mathcal{L}(d)$ et d contient $\mathbf{m}, \mathbf{m}', \mathbf{m}_1, \mathbf{m}_2$. Soit ces messages sont dans le bon ordre dans d pour obtenir les liens entre eux par la projection sur les processus, soit ils sont dans le champ d'application d'une étoile de Kleene, et donc on ne peut voir les réelles dépendances entre eux. Cependant, comme nous avons considéré que les première et dernière itérations de chaque action qui peut être répétée, nous pouvons aussi considérer toutes les dépendances possibles et donc nous trouverons cette composante fortement connexe et en particulier l'arc RS.
  Enfin, nous avons $comp \in \text{COMP}^k_{\mathsf{real}}(\mathcal{S})$, $comp \subseteq \mathsf{GDE}(d^{\circledast})$, $d \in \text{DNF}^k_{\mathsf{real}}(\mathcal{S})$ tel que $comp$ contient un arc RS, et donc l'affirmation B est vraie, ce qui est une contradiction.
- Soit e est K_F-synchronisable, $K > k'$ et cela implique que : soit $\mathsf{GDE}(e)$ contient une composante fortement connexe $comp$ de taille K, $K > k$ et donc la plus grande composante fortement connexe de $\mathsf{GD}(e)$ est plus petite que $comp$. Alors, il existe des arcs dans $\mathsf{GDE}(e)$ qui n'apparaissent pas dans $\mathsf{GD}(e)$. Les arcs ajoutés dans $\mathsf{GDE}(e)$ sont construits à partir des RÈGLES 1 à 5. Les RÈGLES 1 à 4 ajoutent un arc entre deux messages déjà connectés, donc cela ne change pas les composantes fortement connexes. Alors, la plus grande composante fortement connexe dans $\mathsf{GDE}(e)$ contient un arc construit par la RÈGLE 5. On peut déduire qu'il existe un message non couplé $\mathbf{m}$ et un message couplé $\mathbf{m}'$ envoyés au même destinataire. (Comme précédemment, si le système communique en pair à pair, nous avons une contradiction. Nous supposons alors que le système communique en boîte aux lettres.)
  De plus, comme ces deux messages ne peuvent appartenir à la même composante fortement connexe dans $\mathsf{GD}(e)$, nous savons qu'ils appartiennent à deux k-échanges différents, et le message non couplé apparaît avant celui qui est couplé, sans quoi il ne pourrait y avoir d'arc dans cette direction. Alors, il existe $d \in D(\mathcal{S})$ tel que $\exists e' \in \mathcal{L}(d)$, $msc(e') = msc(e)$.

- ✻ soit tous les messages de $comp$ apparaît dans $\mathsf{GDE}(d^{\circledast})$, et donc $comp \in \mathsf{GDE}(d^{\circledast})$ et $comp \in \textsc{Comp}^{k}_{\mathsf{real}}(\mathcal{S})$ alor $k' \geq K$ ce qui est une contradiction ;
- ✻ ou certains messages n'apparaissent pas, ce qui signifie qu'il existe des étoiles de Kleene. On sait que les itérations d'un même message sont liées donc toutes les itérations d'un message appartenant à $comp$ appartiennent à $comp$, en particulier les première et dernière. Alors, comme $\mathsf{GDE}(d^{\circledast})$ contient des sommets représentant les première et dernière itérations d'un message répété, si $comp$ en contient certains, alors les deux itérations apparaissent dans $\mathsf{GDE}(d^{\circledast})$. Alors, l'affirmation A est vraie et nous avons une contradiction.

□

6.4.3 La décidabilité de l'accessibilité dans un système k_{so}-synchronisable

Nous avons donc défini et caractérisé la k_{so}-synchronisabilité. Intéressons-nous à présent aux différents problèmes associés. Dans un premier temps, montrons que l'accessibilité d'un état de contrôle est décidable dans un système k_{so}-synchronisable. Comme nous l'avons fait pour la k-synchronisabilité, nous construisons une relation permettant de relier des k_{so}-échanges. On définit alors la transition $\xrightarrow[\mathsf{so}]{e,k}$ sur des configurations abstraites de la forme $(\vec{\ell}, \vec{bu})$ où $\vec{\ell}$ est un état de contrôle du système et $\vec{bu}$ (pour *boolean unmatched*) un vecteur composé d'un booléen pour chaque canal, signifiant la présence d'un message non couplé dans celui-ci. On note $\vec{bu_0}$ le vecteur initial $(0)_{c \in \vec{c}}$.

On s'assure alors qu'aucun message couplé n'est envoyé dans un canal contenant un message non couplé d'un k_{so}-échange précédent. Il n'est pas nécessaire de le vérifier pour le k_{so}-échange courant, la condition $(\vec{\ell}, \vec{c_\emptyset}) \overset{e}{\Mapsto} (\vec{\ell}', \vec{c})$ nous assure que e est mb-réalisable.

$$\frac{\begin{array}{c} e = a_1 \cdots a_n \in (\mathtt{S} + \mathtt{R})^{\leq 2k} \quad \sharp(e, \mathtt{s}(\_,\_,\_)) \leq k \\ (\vec{\ell}, \vec{c_\emptyset}) \overset{e}{\Mapsto} (\vec{\ell'}, \vec{c}) \text{ pour un certain } \vec{c} \\ \text{pour tout } c \in \vec{c} \text{ avec } c = c_q \text{ si } \mathsf{com} = \mathsf{mb} \text{ et } c = c_{p,q} \text{ si } \mathsf{com} = \mathsf{pp}, \\ \text{si } bu(c) = 1 \text{ alors } \nexists i, j \text{ tels que } a_i \vdash a_j \text{ et } e_i = \mathtt{s}(p,q,m) \\ \text{si } \exists i, a_i = \mathtt{s}(p,q,m) \text{ tel que } \nexists j, a_i \vdash a_j \text{ alors } bu'(c) = 1 \\ \text{sinon } bu'(c) = bu(c) \end{array}}{(\vec{\ell}, \vec{bu}) \xrightarrow[\mathsf{so}]{e,k} (\vec{\ell'}, \vec{bu'})}$$

FIGURE 6.34 – Définition de la relation de transition $\xrightarrow[\mathsf{so}]{e,k}$

Exemple 6.4.3 – La Figure 6.35.a représente un MSC divisé en 2 phases. Nous nous demandons ici si ces phases sont des k_{so}-échanges. Soit $e = e_1 \cdot e_2$ la linéarisation choisie, avec $e_1 = \mathtt{s}(p,q,m_1)$ et $e_2 = \mathtt{s}(p,r,m_2) \cdot \mathtt{s}(p,q,m_4) \cdot \mathtt{r}(p,q,m_4) \cdot \mathtt{s}(r,p,m_3) \cdot \mathtt{r}(r,p,m_3) \cdot \mathtt{r}(p,r,m_2)$. Supposons que $\mathcal{S}$ communique en boîte aux lettres et qu'il existe $\vec{\ell}$, $\vec{\ell}'$ des états de contrôle dans $\mathcal{S}$ tels que $\vec{\ell}_0 \overset{e_1}{\Longrightarrow} \vec{\ell} \overset{e_1}{\Longrightarrow} \vec{\ell}'$. Il paraît évident que e_1 est un k_{so}-échange et que donc il existe $\vec{bu}$ tel que $(\vec{\ell_0}, \vec{bu_0}) \xrightarrow[\mathsf{so}]{e_1,k} (\vec{\ell}, \vec{bu})$ avec $bu(c_q) = 1$ car m_1 n'est pas couplé.

Concentrons-nous à présent sur e_2. On constate que les messages m_2, m_3 et m_4 sont envoyés et reçus dans un ordre cohérent avec la communication en boîte aux lettres, ce qui est confirmé par l'absence de cycle SS dans la deuxième partie du graphe de dépendances étendu, Figure 6.35.b. Donc, on a bien $(\vec{\ell}, \vec{c_\emptyset}) \overset{e_2}{\Longmapsto} (\vec{\ell}', \vec{c})$. Pour tout c, on a $bu'(c) = bu(c)$. Si pour tout $c \in \vec{c}$, $bu(c) = 0$, alors on pourrait conclure que $e_1 \cdot e_2$ est mb-réalisable. Cependant, $bu(c_q) = 1$, le message m_4 est couplé et est destiné au canal c_q donc nous avons une erreur, et on ne peut pas avoir la relation $\xrightarrow[\text{so}]{e_1,k} \xrightarrow[\text{so}]{e_2,k}$.

On peut confirmer cela avec le graphe de dépendances étendu, on constate un cycle SS montrant qu'on ne peut concaténer $msc(e_1)$ et $msc(e_2)$ et obtenir un MSC mb-réalisable.

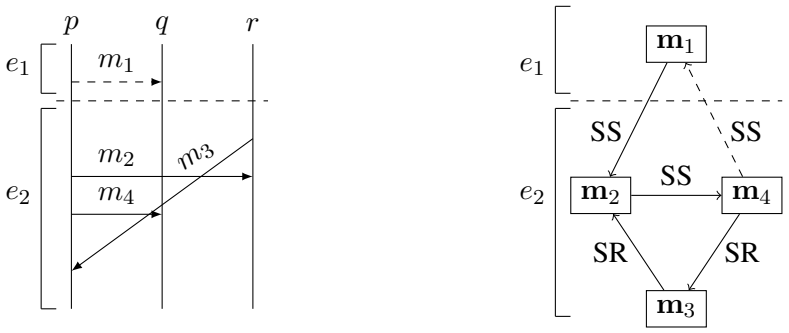


FIGURE 6.35 – Un MSC divisé et son graphe de dépendances étendu

Cette relation de transition $\xrightarrow[\text{so}]{e,k}$ nous permet alors de reconnaître un MSC k_{so}-synchrone.

Lemme 6.4.5. *Un MSC μ est k_{so}-synchrone si et seulement s'il existe une linéarisation $e = e_1 \cdots e_n$ telle que $(\vec{\ell_0}, \overrightarrow{bu_0}) \xrightarrow[\text{so}]{e_1,k} \cdots \xrightarrow[\text{so}]{e_n,k} (\vec{\ell}, \overrightarrow{bu})$.*

Démonstration.

$\Rightarrow$ Comme μ est k_{so}-synchrone alors il existe $e = e_1 \cdots e_n$ telle que e est une linéarisation réalisable de μ. La preuve se fait par récurrence sur n. Posons $\mathsf{GDE}(e) = (V, E)$.

Base Si $n = 1$ alors $e = e_1$. Alors il y a seulement un k_{so}-échange et le graphe de dépendances étendu local $\mathsf{GDE}(e)$ est le même que le graphe complet $\mathsf{GDE}(e)$. Par hypothèse, comme μ est réalisable, on a alors, pour un certain $\vec{c}$, $(\vec{\ell_0}, \vec{c_\emptyset}) \overset{e}{\Longmapsto} (\vec{\ell}, \vec{c})$.
Comme il s'agit du premier k_{so}-échange, $bu(c) = 0$ pour tout $c \in \vec{c}$. Alors il existe $\overrightarrow{bu'}$ tel que $(\vec{\ell_0}, \overrightarrow{bu_0}) \xrightarrow[\text{so}]{e,k} (\vec{\ell}, \overrightarrow{bu'})$.

Récurrence Si $n > 1$, par hypothèse de récurrence, nous avons

$$(\vec{\ell_0}, \overrightarrow{bu_0}) \xrightarrow[\text{so}]{e_1,k} \cdots \xrightarrow[\text{so}]{e_{n-1},k} (\overrightarrow{\ell_{n-1}}, \overrightarrow{bu_{n-1}})$$

Commençons par le cas de la communication en boîte aux lettres.

- ✢ Puisque e est mb-réalisable, $(\overrightarrow{\ell_{n-1}}, \overrightarrow{c_\emptyset}) \overset{e}{\Longmapsto} (\overrightarrow{\ell_n}, \overrightarrow{c})$ pour un certain vecteur $\overrightarrow{c}$.
  Par contradiction, supposons qu'il existe c_q tel que $bu(c_q) = 1$ et tel qu'il existe un message couplé m avec $\mathsf{proc}_{\mathtt{R}}(m) = q$. Comme $bu(c_q) = 1$, par hypothèse de récurrence, il existe $1 \leq i \leq n-1$, tel qu'il existe un message non couplé $m' \in e_i$ avec $\mathsf{proc}_{\mathtt{R}}(m') = q$. Alors $\mathtt{s}(p, q, m') < \mathtt{s}(p', q, m)$ dans e. Or, m est couplé et m' ne l'est pas, donc e ne peut être mb-réalisable, ce qui est une contradiction.

Passons maintenant à la communication en pair à pair.

- ✢ Puisque e est pp-réalisable, $(\overrightarrow{\ell_{n-1}}, \overrightarrow{c_\emptyset}) \overset{e}{\Longmapsto} (\overrightarrow{\ell_n}, \overrightarrow{c})$ pour un certain vecteur $\overrightarrow{c}$.
  Par contradiction, supposons qu'il existe $c_{p,q}$ tel que $bu(c_{p,q}) = 1$ et tel qu'il existe un message couplé m avec $\mathsf{proc}_{\mathtt{S}}(m) = p$ et $\mathsf{proc}_{\mathtt{R}}(m) = q$. Comme $bu(c_{p,q}) = 1$, par hypothèse de récurrence, il existe $1 \leq i \leq n-1$, tel qu'il existe un message non couplé $m' \in e_i$ avec $\mathsf{proc}_{\mathtt{S}}(m') = p$ et $\mathsf{proc}_{\mathtt{R}}(m') = q$. Alors $\mathtt{s}(p, q, m') < \mathtt{s}(p, q, m)$ dans e. Or, m est couplé et m' ne l'est pas, donc e ne peut être pp-réalisable, ce qui est une contradiction.

$\Leftarrow$ Soit $e = e_1 \cdots e_n$ telle que $(\overrightarrow{\ell_0}, \overrightarrow{bu_0}) \xrightarrow[\mathsf{so}]{e_1,k} \cdots \xrightarrow[\mathsf{so}]{e_n,k} (\overrightarrow{\ell'}, \overrightarrow{bu})$.

Pour tout $1 \leq i \leq n$, $(\overrightarrow{\ell_{i-1}}, \overrightarrow{c_\emptyset}) \overset{e_i}{\Longmapsto} (\overrightarrow{\ell_i}, \overrightarrow{c_i})$ pour un certain $\overrightarrow{c_i}$ et chaque e_i contient au plus k envois donc chaque e_i est un k_{so}-échange valide.

Supposons par contradiction que e n'est pas k_{so}-synchronisable. Alors, e n'est pas réalisable. Commençons par la communication en boîte aux lettres.

- ✢ Supposons que e ne soit pas mb-réalisable. On sait que deux messages ne peuvent pas être envoyés et reçus dans deux ordres différents car soit ils appartiennent au même k_{so}-échange ce qui ne validerait alors pas $(\overrightarrow{\ell_{i-1}}, \overrightarrow{c_\emptyset}) \overset{e_i}{\Longmapsto} (\overrightarrow{\ell_i}, \overrightarrow{c_i})$, soit ils sont dans deux k_{so}-échanges différents mais alors une réception est séparée de son envoi, ce n'est pas possible dans un k_{so}-échange. Alors, il existe deux messages dans deux k_{so}-échanges différents tels que $m \in e_i$, $m' \in e_j$, $i < j$, $\mathsf{proc}_{\mathtt{R}}(m) = \mathsf{proc}_{\mathtt{R}}(m') = q$ avec m non couplé et m' couplé. Alors, comme m est non couplé, alors $(\overrightarrow{\ell_{i-1}}, \overrightarrow{bu_{i-1}}) \xrightarrow[\mathsf{so}]{e_i,k} (\overrightarrow{\ell_i}, \overrightarrow{bu_i})$ avec $bu_i(c_q) = 1$. Par hypothèse, on a $(\overrightarrow{\ell_{j-1}}, \overrightarrow{bu_{j-1}}) \xrightarrow[\mathsf{so}]{e_j,k} (\overrightarrow{\ell_j}, \overrightarrow{bu_j})$. Or, comme $i < j$, $bu_{j-1}(c_q) = 1$ donc on ne peut pas avoir un envoi couplé vers q dans e_j. Ceci mène donc à une contradiction et e est k_{so}-synchronisable.

Passons à la communication en pair à pair.

- ✢ Supposons que e ne soit pas pp-réalisable. Comme précédemment, on sait qu'on ne peut pas avoir deux messages couplés qui violent la réalisabilité. Alors, il existe deux messages dans deux k_{so}-échanges différents tels que $m \in e_i$, $m' \in e_j$, $i < j$, $\mathsf{proc}_{\mathtt{S}}(m) = \mathsf{proc}_{\mathtt{S}}(m') = p$, $\mathsf{proc}_{\mathtt{R}}(m) = \mathsf{proc}_{\mathtt{R}}(m') = q$ avec m non couplé et m' couplé. Alors, comme m est non couplé, alors $(\overrightarrow{\ell_{i-1}}, \overrightarrow{bu_{i-1}}) \xrightarrow[\mathsf{so}]{e_i,k} (\overrightarrow{\ell_i}, \overrightarrow{bu_i})$ avec $bu_i(c_{p,q}) = 1$. Par hypothèse, on a $(\overrightarrow{\ell_{j-1}}, \overrightarrow{bu_{j-1}}) \xrightarrow[\mathsf{so}]{e_j,k} (\overrightarrow{\ell_j}, \overrightarrow{bu_j})$. Or, comme $i < j$, $bu_{j-1}(c_{p,q}) = 1$ donc on ne peut pas avoir un envoi couplé vers q par p dans e_j. Ceci mène donc à une contradiction et e est k_{so}-synchronisable.

□

Ainsi, les états de contrôle du système et les vecteurs de booléens pour chaque canal sont finis. Alors, des configurations abstraites de cette forme existent en nombre fini dans un système et nous pouvons alors conclure qu'un tel système permet la décidabilité d'accessibilité d'un état de contrôle.

Théorème 6.4.6. *Soit $\mathcal{S}$ un système k_{so}-synchronisable et $\vec{\ell}$ un état de contrôle de $\mathcal{S}$. Savoir si $\vec{\ell}$ est accessible dans $\mathcal{S}$ est décidable.*

Démonstration.

Il existe un nombre fini de configurations abstraites de la forme $(\vec{\ell}, \vec{bu})$ avec $\vec{\ell}$ un état de contrôle global de $\mathcal{S}$ et $\vec{bu} \in [0,1]^{\mathbb{P}}$ dans un cas en boîte aux lettres, ou $\vec{bu} \in [0,1]^{\mathbb{P}\times\mathbb{P}}$ dans un cas en pair à pair. Alors, $\xrightarrow[\mathsf{so}]{\cdot,k}$ est une relation sur un ensemble fini, et l'ensemble des MSC k_{so}-synchrones $\text{MSC}_k^{so}(\mathcal{S})$ du système $\mathcal{S}$ forme un langage régulier. Ainsi, il est décidable de savoir si une configuration abstraite donnée de la forme $(\vec{\ell}, \vec{bu})$ est accessible depuis la configuration initiale suivant une exécution k_{so}-synchronisable étant une linéarisation réalisable d'un MSC contenu dans $\text{MSC}_k^{so}(\mathcal{S})$.

□

6.4.4 La décidabilité de la k_{so}-synchronisabilité - Avec paramètre k

Passons à présent au problème de l'appartenance à la classe des systèmes k_{so}-synchronisables pour un k et un système donnés. Nous adaptons la preuve de la décidabilité d'appartenance pour un k donné à la classe des systèmes k-synchronisables à la k_{so}-synchronisabilité, décrite dans le Chapitre 4. Les différences résideront dans le fait que la division en k_{so}-échanges doit être une exécution réalisable du système. Nous n'auront plus à vérifier les liens causaux entre les messages contenus dans les k-échanges précédents et ceux du k-échange courant : nous savons qu'ils ont forcément été envoyé avant. Pour vérifier que le MSC résultant est bien réalisable, il faudra alors seulement vérifier que les messages du k-échange courant ont pu être envoyés après ceux des k-échanges précédents. L'abstraction des canaux est alors simplifiée, nous ne retenons plus qu'une information : quels canaux contiennent au moins un message non couplé.

L'objectif est alors de définir un système instrumenté $\mathcal{S}'$ afin de reconnaître à la fois les exécutions k_{so}-synchronisables du système $\mathcal{S}$ mais également les exécutions critiques si elles existent. Pour rappel, une exécution critique est une exécution qui n'est pas k_{so}-synchronisable, mais qui le devient si l'on supprime la dernière réception : tout système qui n'est pas k_{so}-synchronisable admet une telle exécution.

Définition 6.4.4 (Exécution critique)**.** Soit $e \in \text{EX}(\mathcal{S})$ une exécution de $\mathcal{S}$. e est critique pour k si et seulement si e n'est pas k_{so}-synchronisable, $e = e' \cdot r$ telle que $e' \in (\mathtt{S} + \mathtt{R})^*$, $r \in \mathtt{R}$ et e' est k_{so}-synchronisable.

Ce système instrumenté est identique à celui défini dans le Chapitre 4, Définition 4.3.2. Pour rappel, il contient un processus supplémentaire π capable de dévier la réception d'un message et donc de considérer les exécutions critiques.

Pour toute exécution $e' \cdot r \in \text{EX}(\mathcal{S})$ qui termine donc par une réception, il existe une exécution $\mathsf{deviate}(e' \cdot r) \in \text{EX}(\mathcal{S}')$ où le message associé à la réception r est dévié par π. Formellement, si $e = e_1 \cdot s \cdot e_2 \cdot r$ avec $r = \mathtt{r}(p,q,m)$ et $s \vdash\!\dashv r$ alors :

$$\mathsf{deviate}(e' \cdot r) = e_1 \cdot \mathtt{s}(p,\pi,(q,m)) \cdot \mathtt{r}(p,\pi,(q,m)) \cdot e_2 \cdot \mathtt{s}(\pi,q,m) \cdot \mathtt{r}(\pi,q,m)$$

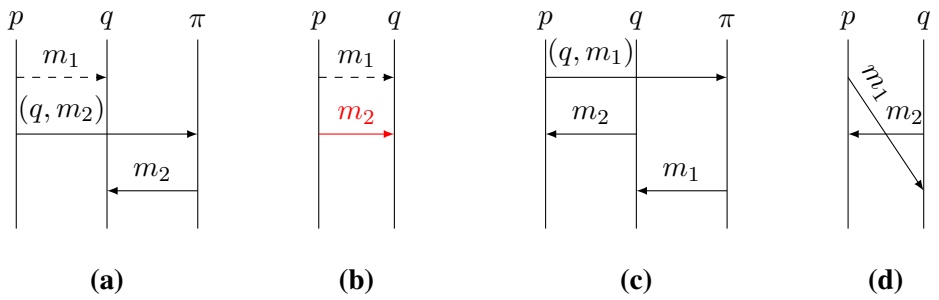


FIGURE 6.36 – Des exécutions de $\mathcal{S}'$ et leurs correspondances dans $\mathcal{S}$

S'il existe une exécution critique dans $\mathcal{S}$, il existe une exécution faisable et mauvaise correspondante dans le système instrumenté $\mathcal{S}'$. Ci-dessous, les définitions d'exécutions faisable et mauvaise, identiques dans les contextes de la k-synchronisabilité et de la k_{so}-synchronisabilité, sont rappelées.

Définition 6.4.5 (Exécution faisable, exécution mauvaise)**.** Une exécution k_{so}-synchronisable $e' \in \text{Ex}(\mathcal{S}')$ est faisable s'il existe une exécution $e{\cdot}r \in \text{Ex}(\mathcal{S})$ tel que $\mathsf{deviate}(e{\cdot}r) = e'$. Une exécution faisable $e' = \mathsf{deviate}(e \cdot r) \in \text{Ex}(\mathcal{S}')$ est mauvaise si l'exécution $e \cdot r$ n'est pas k_{so}-synchronisable dans $\mathcal{S}$.

Revoyons un exemple d'exécutions faisables et mauvaises.

Exemple 6.4.4 – La Figure 6.36.a représente le MSC $msc(e')$ tel que $e'_1 = \mathtt{s}(p,q,m_1) \cdot \mathtt{s}(p,\pi,(q,m)) \cdot \mathtt{r}(p,\pi,(q,m)) \cdot \mathtt{s}(\pi,q,m_2) \cdot \mathtt{r}(\pi,q,m_2)$. Soit $e_1 \cdot r_1$ la séquence d'actions telle que $\mathsf{deviate}(e_1 \cdot r_1) = e'_1$. On a alors $e_1 \cdot r_1 = \mathtt{s}(p,q,m_1) \cdot \mathtt{s}(p,q,m_2) \cdot \mathtt{r}(p,q,m_2)$, et le MSC de la Figure 6.36.b est le MSC correspondant. On constate que le message m_2 pose problème : il ne peut être couplé étant donné que le message m_1 est déjà stocké dans le canal $c_{p,q}$ si $\mathsf{com} = \mathsf{pp}$, ou dans le canal c_q si $\mathsf{com} = \mathsf{mb}$. Ainsi, $e_1 \cdot r_1$ n'est pas ni mb-réalisable, ni pp-réalisable. Ainsi, quelque soit la communication du système, e'_1 n'est pas faisable.

Voyons un autre cas. Le MSC de la Figure 6.36.c correspond à l'exécution $e'_2 = \mathtt{s}(p,\pi,(q,m_1)){\cdot}\mathtt{r}(p,\pi,(q,m_1)) \cdot \mathtt{s}(q,p,m_2) \cdot \mathtt{r}(q,p,m_2) \cdot \mathtt{s}(\pi,q,m_1) \cdot \mathtt{r}(\pi,q,m_1)$. On constate que $e'_2 = \mathsf{deviate}(e_2 \cdot r_2)$ avec $e_2 \cdot r_2 = \mathtt{s}(p,q,m_1) \cdot \mathtt{s}(q,p,m_2) \cdot \mathtt{r}(q,p,m_2) \cdot \mathtt{r}(q,p,m_1)$. $msc(e_2 \cdot r_2)$ est représenté dans la Figure 6.36.d. Ce MSC est mb-réalisable (et pp-réalisable) donc, quelque soit la communication de $\mathcal{S}$, e'_2 est faisable. D'autre part, si l'on pose $k = 1$, alors $e_2 \cdot r_2$ n'est pas k_{so}-synchronisable, alors que e_2 est k_{so}-synchronisable. Ainsi, $e_2 \cdot r_2$ constitue une exécution critique pour $k = 2$. De plus, e'_2 est alors une exécution mauvaise, car elle reconnaît une exécution critique.

Savoir si un système est k_{so}-synchronisable pour un k donné peut se réduire à la recherche d'une exécution faisable et mauvaise, car la présence d'une exécution mauvaise dans $\mathcal{S}'$ signifie qu'elle ne peut pas être k_{so}-synchronisable dans $\mathcal{S}$. Si celle-ci est également faisable, alors nous sommes certains que son équivalent existe dans $\mathcal{S}$ et donc qu'il existe une exécution qui n'est pas k_{so}-synchronisable dans $\mathcal{S}$, et ainsi que $\mathcal{S}$ n'est pas k_{so}-synchronisable.

Lemme 6.4.7. *Un système $\mathcal{S}$ n'est pas k_{so}-synchronisable si et seulement s'il existe une exécution e' de $\mathcal{S}'$ qui est faisable et mauvaise.*

Démonstration.

⇒ Soit $\mathcal{S}$ un système non k_{so}-synchronisable. Alors il existe une exécution critique $e{\cdot}r \in \text{EX}(\mathcal{S})$ avec $r = \mathtt{r}(p, q, m)$, telle que e est k_{so}-synchronisable et $e \cdot r$ n'est pas k_{so}-synchronisable.

Soit $e' = \mathsf{deviate}(e \cdot r)$ telle que $e' \in \text{EX}(\mathcal{S}')$. Alors, par définition, comme $e \cdot r \in \text{EX}(\mathcal{S})$, e' est faisable. De plus, e est k_{so}-synchronisable, alors e' qui dévie et donc sépare la réception r de son envoi, est k_{so}-synchronisable aussi. Donc e' est par définition mauvaise.

⇐ S'il existe e' une exécution faisable et mauvaise alors $e' = \mathsf{deviate}(e \cdot r)$ avec $e \cdot r \in \text{EX}(\mathcal{S})$ telles que e' est k_{so}-synchronisable et $e \cdot r$ ne l'est pas. Alors, il existe une exécution dans $\mathcal{S}$ qui n'est pas k_{so}-synchronisable et donc $\mathcal{S}$ n'est pas k_{so}-synchronisable.

□

Reconnaissance des exécutions faisables

Tout comme précédemment, nous construisons une relation de transition permettant de reconnaître les exécutions faisables. Dans celle-ci, nous vérifions que le k_{so}-échange observé est faisable dans le système d'origine en tenant compte des informations conservées des k_{so}-échanges précédents. La transition $\xRightarrow[\mathsf{so{\cdot}feas}]{e,k}$ est définie sur des configurations abstraites de la forme $(\vec{\ell}, \vec{bu}, \mathsf{c_{dest}})$ où $\vec{\ell}$ est un état de contrôle de $\mathcal{S}'$, $\vec{bu}$ est un vecteur composé d'un booléen pour chaque canal, indiquant la présence d'un message non couplé dans celui-ci, $\mathsf{c_{dest}}$ se souvient du canal qui aurait dû recevoir le message dévié s'il avait été envoyé au destinataire originel.

Nous regarderons en parallèle les systèmes communicquant en boîte aux lettres et ceux communiquant en pair à pair. Pour vérifier que la suite de k_{so}-échanges observée est mb-réalisable dans le système d'origine, il faut s'assurer d'une part qu'après l'envoi d'un message destiné à q au processus π, aucun message couplé n'est envoyé à q, par n'importe quel expéditeur, et d'autre part qu'aucun message non couplé n'a été envoyé à q avant le message dévié. Cette dernière vérification sera faite en observant tous les canaux, et en transmettant le fait qu'ils contiennent un ou plusieurs messages non couplés, k_{so}-échange après k_{so}-échange. En effet, on ne sait pas au début de l'exécution quel sera le canal concerné par la déviation. Dans le même but, pour savoir si une suite de k_{so}-échanges est pp-réalisable, il faudra s'assurer qu'après l'envoi d'un message destiné à q envoyé à π, aucun message couplé n'est envoyé à q par l'expéditeur du message dévié, et qu'aucun message non couplé n'a été envoyé par p à q avant la déviation.

Lemme 6.4.8. *Une exécution $e' = e_1 \cdot \mathtt{s}(p, \pi, (q, m)) \cdot \mathtt{r}(p, \pi, (q, m)) \cdot e_2 \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$ n'est pas faisable si et seulement s'il existe*

- ✢ *soit un message non couplé m' tel que $\mathtt{s}(p', q, m') \in e_1$*
- ✢ *soit un message couplé m' tel que $\mathtt{s}(p', q, m') \in e_2$*

avec $p' = p$ si com = pp.

Démonstration.

⇒ Soit $e' = e_1 \cdot \mathtt{s}(p, \pi, (q, m)) \cdot \mathtt{r}(p, \pi, (q, m)) \cdot e_2$ telle que $e' \in \text{EX}(\mathcal{S}')$ n'est pas faisable. Posons $e \cdot r$ tel que $\mathsf{deviate}(e \cdot r) = e'$. Ainsi, comme e' n'est pas faisable, $e \cdot r \notin \text{EX}(\mathcal{S})$.

Or e_1 ne concerne que des processus et des actions qui existent également dans $\mathcal{S}$ donc $e_1 \in \text{EX}(\mathcal{S})$. De la même façon, si les deux messages posant problème étaient dans e_2, e' ne serait pas réalisable non plus. Ainsi, le message m est concerné et alors, soit il existe un message m' non couplé dans e_1, ou bien couplé dans e_2 tel que $\mathsf{proc}_{\mathtt{R}}(m) = \mathsf{proc}_{\mathtt{R}}(m')$ et, si $\mathsf{com} = \mathsf{pp}$ alors $\mathsf{proc}_{\mathtt{S}}(m) = \mathsf{proc}_{\mathtt{S}}(m')$.

$\Leftarrow$ Soit $e' = e_1 \cdot \mathtt{s}(p, \pi, (q, m)) \cdot \mathtt{r}(p, \pi, (q, m)) \cdot e_2 \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$. Alors, il existe une séquence d'action $e \cdot r$ telle que $e' = \mathsf{deviate}(e \cdot r)$ et donc $e = e_1 \cdot \mathtt{s}(p, q, m) \cdot e_2 \cdot \mathtt{r}(p, q, m)$.

Supposons qu'il existe un message non couplé m' tel que $\mathtt{s}(p', q, m') \in e_1$, tel que $p' = p$ si $\mathsf{com} = \mathsf{pp}$. Alors, il existe un message non couplé envoyé avant le message m qui est couplé, dans le même canal. $e \cdot r$ n'est pas donc pas réalisable.

Supposons maintenant qu'il existe un message couplé m' tel que $\mathtt{s}(p', q, m') \in e_2$ tel que $p' = p$ si $\mathsf{com} = \mathsf{pp}$. Alors, il existe deux messages m et m' couplés, envoyés dans le même canal dans un certain ordre, mais lus dans l'ordre inverse. Donc $e \cdot r$ n'est pas réalisable.

Ainsi, $e \cdot r$ n'est pas réalisable donc e' n'est pas faisable.

□

Une fois que la déviation vers π a eu lieu, il est nécessaire de savoir quel était le canal qui devait recevoir le message à l'origine. C'est l'objectif de la variable $\mathsf{c}_{\mathsf{dest}}$ qui est initialisé à $\perp$ et mise à jour lors de l'envoi à π. Tant que $\mathsf{c}_{\mathsf{dest}} = \perp$, on sait que l'envoi à π n'a pas eu lieu dans un k_{so}-échange précédent. Lorsque l'envoi à π a lieu dans le k_{so}-échange courant, la variable env permet d'identifier l'indice de cette action. On peut alors vérifier qu'il n'y a pas eu de messages non couplés concernant $\mathsf{c}_{\mathsf{dest}}$ dans ce k_{so}-échange ou dans un k_{so}-échange précédent et seuls les messages couplés envoyés dans $\mathsf{c}_{\mathsf{dest}}$ après cela seront problématiques.

$$e = a_1 \cdots a_n \in (\mathtt{S} + \mathtt{R})^{\leq 2k} \quad (1)$$

$$(\vec{\ell}, \vec{bu}) \xRightarrow[\mathsf{so}]{e,k} (\vec{\ell}', \vec{bu}') \quad (2)$$

$$(\forall \mathbf{m}, \mathbf{m}')\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \mathsf{proc}_{\mathtt{R}}(\mathbf{m}') = \pi \implies \mathbf{m} = \mathbf{m}' \wedge \mathsf{c}_{\mathsf{dest}} = \perp \quad (3)$$

$$(\nexists \mathbf{m})\mathsf{proc}_{\mathtt{R}}(\mathbf{m}) = \pi \implies \mathsf{env} = \perp \quad (4)$$

$$(\forall \mathbf{m})\mathbf{m} \ni a_i = \mathtt{s}(p, \pi, (q, m)) \implies \mathsf{c}'_{\mathsf{dest}} = c_1 \wedge \mathsf{env} = i \wedge bu(\mathsf{c}'_{\mathsf{dest}}) = 0 \quad (5)$$

$$(\forall \mathbf{m})\mathbf{m} = \{a_j = \mathtt{s}(p', q', m')\} \wedge \mathsf{env} \neq \perp \implies c_2 \neq \mathsf{c}'_{\mathsf{dest}} \vee j > \mathsf{env} \quad (6)$$

$$(\forall \mathbf{m})\mathbf{m} = \{a_j = \mathtt{s}(p', q', m'), a_{j'}\} \implies$$
$$(\mathsf{env} \neq \perp \wedge (c_2 \neq \mathsf{c}'_{\mathsf{dest}} \vee j < \mathsf{env})) \vee (\mathsf{c}_{\mathsf{dest}} \neq \perp \wedge c_2 \neq \mathsf{c}_{\mathsf{dest}}) \quad (7)$$

$$\frac{}{(\vec{\ell}, \vec{bu}, \mathsf{c}_{\mathsf{dest}}) \xRightarrow[\mathsf{so\cdot feas}]{e,k} (\vec{\ell}', \vec{bu}', \mathsf{c}'_{\mathsf{dest}})}$$

FIGURE 6.37 – Définition de la relation de transition $\xRightarrow[\mathsf{so\cdot feas}]{e,k}$

Dans la Figure 6.37, $c_1 = c_q, c_2 = c_{q'}$ si $\mathsf{com} = \mathsf{mb}$, ou bien $c_1 = c_{p,q}, c_2 = c_{p',q'}$ si $\mathsf{com} = \mathsf{pp}$.

Plus précisément, les conditions (1) et (2) s'assurent que e soit une séquence d'au plus $2k$ actions et qu'il s'agit d'un k_{so}-échange. La condition (3) vérifie qu'il n'existe qu'un seul envoi à π dans e associé aux k_{so}-échanges précédents. La condition (4) définit la valeur de env s'il n'existe pas d'envoi à π. La condition (5) définit les valeurs des variables env et $\mathsf{c}_{\mathsf{dest}}$ lorsque l'envoi à

π a lieu. Elle vérifie également qu'aucun message non couplé n'est présent dans le canal $\mathsf{c}'_{\mathsf{dest}}$ au début du k_{so}-échange. La condition (6) vérifie qu'il n'en existe pas non plus depuis le début du k_{so}-échange jusqu'à l'envoi à π. Enfin, la condition (7) vérifie que pour tout message couplé, celui-ci ne crée pas d'erreur : s'il appartient au k_{so}-échange de la déviation, il doit soit être envoyé avant le message dévié, soit ne pas concerner le canal $\mathsf{c}'_{\mathsf{dest}}$, s'il appartient à un k_{so}-échange subséquent à celui de la déviation, on s'assure seulement qu'il ne concerne pas le canal $\mathsf{c}'_{\mathsf{dest}}$.

Exemple 6.4.5 – Soit $msc(e_1 \cdot e_2 \cdot e_3)$ le MSC de la Figure 6.38 et supposons que l'on a $\vec{\ell}$ tel que $(\vec{\ell_0}, \vec{c_\emptyset}) \overset{e_1}{\Longmapsto} (\vec{\ell}, \vec{c}) \overset{e_2}{\Longmapsto} (\vec{\ell'}, \vec{c'})$ dans un système communiquant en boîte aux lettres. On peut établir que $(\vec{\ell_0}, (0,0), \bot) \xRightarrow[\mathsf{so{\cdot}feas}]{e_1,1} (\vec{\ell}, (0,0), c_q)$. Cependant, le message m_2 rend la condition (6) fausse, car le message m_2 est couplé, or $\mathsf{env} = \bot$, $\mathsf{c}_{\mathsf{dest}} \neq \bot$ mais $\mathsf{c}_{\mathsf{dest}} = c_q$ avec $q = \mathsf{proc}_{\mathtt{R}}(m_2)$. Donc, $e_1 \cdot e_2 \cdot e_3$ n'est pas faisable.

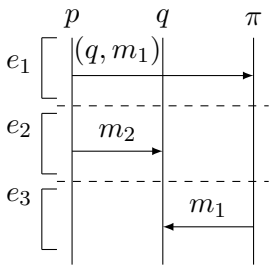


FIGURE 6.38 – Un MSC du système instrumenté divisé en 3 k_{so}-échanges

Lemme 6.4.9. *Soit e' une exécution k_{so}-synchronisable de $\mathcal{S}'$. Alors e' est une exécution faisable si et seulement si $\exists e'' = e_1 \cdots e_n \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$, $msc(e'') = msc(e')$ avec $\vec{bu} \in ([0,1])_{c \in \vec{c}}$, $\mathsf{c}_{\mathsf{dest}} \in \vec{c}$ et un état de contrôle $\vec{\ell}$ tels que*

$$(\vec{\ell_0}, \vec{bu_0}, \bot) \xRightarrow[\mathsf{so{\cdot}feas}]{e_1,k} \cdots \xRightarrow[\mathsf{so{\cdot}feas}]{e_n,k} (\vec{\ell}, \vec{bu}, \mathsf{c}_{\mathsf{dest}})$$

Démonstration.

$\Rightarrow$ Soit $e' \in \text{EX}(\mathcal{S}')$ une exécution k_{so}-synchronisable et faisable. Sans perte de généralités, supposons que e' soit la division en k_{so}-échanges de $msc(e')$. Donc, $e' = e_1 \cdots e_n \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$ tel que chaque e_i est un k_{so}-échange, donc les conditions (1) et (2) sont respectées.

e' est faisable donc il existe une exécution $e \cdot r \in \text{EX}(\mathcal{S})$ tel que $e' = \mathsf{deviate}(e \cdot r)$.

Par récurrence, montrons que pour tout $1 \leq i \leq n$, il existe $\vec{\ell_i}$, $\vec{bu_i}$ et $\mathsf{c}_{\mathsf{dest}i}$ tel que

$$(\vec{\ell_{i-1}}, \vec{bu_{i-1}}, \mathsf{c}_{\mathsf{dest}i-1}) \xRightarrow[\mathsf{so{\cdot}feas}]{k,e_i} (\vec{\ell_i}, \vec{bu_i}, \mathsf{c}_{\mathsf{dest}i})$$

Par la définition de $\mathsf{deviate}(\cdot)$, il n'existe qu'un message envoyé à π dans e', donc la condition (3) est respectée.

Base $n = 1$ alors $e' = e_1 \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$. Posons $a_i = \mathtt{s}(p, \pi, (q, m))$. Alors, $\mathsf{env} = i$, et

- $\mathsf{c}'_{\mathsf{dest}} = c_q$ si com = mb
- $\mathsf{c}'_{\mathsf{dest}} = c_{p,q}$ si com = pp.

Par l'absurde, supposons que la condition (4) n'est pas vérifiée. Alors, $bu_{i-1}(\mathsf{c}'_{\mathsf{dest}}) \neq 0$, ce qui n'est pas possible, car $\overrightarrow{bu} = \overrightarrow{bu_0}$.

Supposons par l'absurde que la condition (5) n'est pas vérifiée, alors il existe un message non couplé dans e_1 envoyé dans $\mathsf{c}'_{\mathsf{dest}}$ avant m. Donc d'après le Lemme 6.4.8, e' n'est pas faisable, ce qui est une contradiction, donc la condition (5) est vérifiée.

Supposons par l'absurde que la condition (6) n'est pas vérifiée. Alors il existe un message couplé m' envoyé dans $\mathsf{c}'_{\mathsf{dest}}$ après l'envoi de m. D'après le Lemme 6.4.8, e' n'est pas faisable, ce qui est une contradiction et assure que la condition (6) est vérifiée.

Toutes les conditions sont respectées et il existe donc $\overrightarrow{\ell}, \overrightarrow{bu}, \mathsf{c}_{\mathsf{dest}}$ tels que $(\overrightarrow{\ell_0}, \overrightarrow{bu_0}, \mathsf{c}_{\mathsf{dest}}) \xRightarrow[\text{so·feas}]{e_1,k} (\overrightarrow{\ell}, \overrightarrow{bu}, \mathsf{c}'_{\mathsf{dest}})$.

Récurrence Supposons que $(\overrightarrow{\ell_0}, \overrightarrow{bu_0}, \mathsf{c}_{\mathsf{dest}}) \xRightarrow[\text{so·feas}]{e_1,k} \cdots \xRightarrow[\text{so·feas}]{e_{n-1},k} (\overrightarrow{\ell_{n-1}}, \overrightarrow{bu_{n-1}}, \mathsf{c}_{\mathsf{dest}n-1})$.

Montrons que $(\overrightarrow{\ell_{n-1}}, \overrightarrow{bu_{n-1}}, \mathsf{c}_{\mathsf{dest}n-1}) \xRightarrow[\text{so·feas}]{e_n,k} (\overrightarrow{\ell_n}, \overrightarrow{bu_n}, \mathsf{c}_{\mathsf{dest}n})$.

Soit $a_i = \mathtt{s}(p, \pi, (q, m)) \in e_j$, $1 \leq i \leq \mid e_j \mid$, $1 \leq j \leq n$. Deux cas sont possibles :

- $j < n$, alors $\mathsf{c}_{\mathsf{dest}n-1} \neq \perp$ et env $= \perp$. Les conditions (4) et (5) sont forcément respectées, car il n'y a pas d'envoi à π dans e_n, et env n'est pas défini.
  Par l'absurde, supposons que la condition (6) n'est pas vérifiée. Alors il existe un message couplé m' envoyé dans $\mathsf{c}_{\mathsf{dest}n-1}$. Alors, d'après le Lemme 6.4.8, e' n'est pas faisable, ce qui est une contradiction. Donc la condition (6) est vérifiée.
- $j = n$, alors $\mathsf{c}_{\mathsf{dest}n-1} = \perp$ et env $= i$. Par l'absurde, supposons que la condition (4) n'est pas respectée. Alors, $bu_{n-1}(\mathsf{c}_{\mathsf{dest}n}) \neq 0$. Alors un message non couplé est envoyé dans $\mathsf{c}_{\mathsf{dest}n-1}$ dans e_1 donc e' n'est pas faisable d'après le Lemme 6.4.8, ce qui est une contradiction, et donc la condition (4) est respectée.
  Par l'absurde, supposons que la condition (5) n'est pas vérifiée. Alors, il existe un message non couplé m' envoyé dans $\mathsf{c}_{\mathsf{dest}n}$ avant m dans e_n. Alors, un message non couplé est envoyé dans $\mathsf{c}_{\mathsf{dest}n-1}$ dans e_1 donc e' n'est pas faisable d'après le Lemme 6.4.8, ce qui est une contradiction, et donc la condition (5) est respectée.
  Par l'absurde, supposons que la condition (6) n'est pas respectée. Alors, il existe un message couplé envoyé après m dans le canal $\mathsf{c}_{\mathsf{dest}n}$ donc, d'après le Lemme 6.4.8, e' n'est pas faisable et l'on atteint une contradiction donc la condition (6) est respectée.

Ainsi, dans tous les cas, toutes les conditions sont respectées et

$$(\overrightarrow{\ell_{n-1}}, \overrightarrow{bu_{n-1}}, \mathsf{c}_{\mathsf{dest}n-1}) \xRightarrow[\text{so·feas}]{e_n,k} (\overrightarrow{\ell_n}, \overrightarrow{bu_n}, \mathsf{c}_{\mathsf{dest}n})$$

ce qui conclut cette partie de la preuve.

$\Leftarrow$ Soit $e' = e_1 \cdots e_n \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$ telle qu'il existe $\overrightarrow{\ell}, \overrightarrow{bu}$ et $\mathsf{c}_{\mathsf{dest}}$ et

$$(\overrightarrow{\ell_0}, \overrightarrow{bu_0}, \perp) \xRightarrow[\text{so·feas}]{e_1,k} \cdots \xRightarrow[\text{so·feas}]{e_n,k} (\overrightarrow{\ell}, \overrightarrow{bu}, \mathsf{c}_{\mathsf{dest}})$$

Les conditions (1) et (2) assurent que e' est composé de k_{so}-échanges et que donc e' est k_{so}-synchronisable. D'après (3), il n'existe qu'un seul message envoyé à π dans e'. Donc il existe $e \cdot r$

telle que $e' = \mathsf{deviate}(e \cdot r)$. Montrons que $e \cdot r \in \mathrm{EX}(\mathcal{S})$, c'est-à-dire que $e \cdot r$ est pp-réalisable ou mb-réalisable selon la communication de $\mathcal{S}$.

Par l'absurde, supposons que $e \cdot r$ n'est pas réalisable, alors il existe soit un message non couplé envoyé avant m soit un message couplé envoyé après, dans le canal d'origine du message dévié, et donc, d'après le Lemme 6.4.8, e' n'est pas faisable, ce qui est une contradiction.

□

Reconnaissance d'une mauvaise exécution

Pour trouver une exécution critique de $\mathcal{S}$, on est donc à la recherche d'une exécution faisable et mauvaise dans le système instrumenté. Nous sommes à présent capables de reconnaître une exécution faisable, il nous faut maintenant reconnaître une mauvaise exécution. Une mauvaise exécution correspond à une exécution qui n'est pas k_{so}-synchronisable dans $\mathcal{S}$, c'est-à-dire que le graphe de dépendances étendu, si la communication est en boîte aux lettres, ou le graphe de dépendances si la communication est en pair à pair, contient une composante fortement connexe de taille supérieure à k. En effet, nous n'avons pas besoin de vérifier l'absence d'arc RS comme précédemment, celui-ci certifiant que l'on peut organiser tous les envois suivis de toutes les réceptions, ce qui est inutile ici.

On rappelle, pour le graphe de dépendances étendu $\mathsf{GDE}(e') = (E, V)$, que les ensembles de messages $\mathsf{Post}^*(\mathbf{m}) = \{\mathbf{m}' \in V \mid \mathbf{m} \to^* \mathbf{m}'\}$ et $\mathsf{Pre}^*(\mathbf{m}) = \{\mathbf{m} \in V \mid \mathbf{m}' \to^* \mathbf{m}\}$ sont les messages accessibles et co-accessibles depuis $\mathbf{m}$ dans le graphe de dépendances, ignorant les arcs étendus. On définit de la même façon les ensembles $\mathsf{EPost}^*(\mathbf{m}) = \{\mathbf{m}' \in V \mid \mathbf{m} \dashrightarrow^* \mathbf{m}'\}$ et $\mathsf{EPre}^*(\mathbf{m}) = \{\mathbf{m} \in V \mid \mathbf{m}' \dashrightarrow^* \mathbf{m}\}$ des messages accessibles et co-accessibles depuis $\mathbf{m}$ dans le graphe de dépendances étendu, comprenant donc tous les arcs.

La taille de la composante fortement connexe à vérifier dans le graphe concerné correspond à l'intersection des messages accessibles depuis $\mathbf{m}_{\mathsf{start}}$ et co-accessibles depuis $\mathbf{m}_{\mathsf{stop}}$.

Lemme 6.4.10. *Une exécution faisable e' est mauvaise si et seulement si*

- *l'ensemble $\mathsf{EPost}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{EPre}^*(\mathbf{m}_{\mathsf{stop}})$ est de taille supérieure ou égale à $k+2$ si la communication est en boîte aux lettres,*
- *l'ensemble $\mathsf{Post}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{Pre}^*(\mathbf{m}_{\mathsf{stop}})$ est de taille supérieure ou égale à $k+2$ si la communication est en pair à pair,*

pour $\mathbf{m}_{\mathsf{start}} \ni \mathtt{s}(p, \pi, (q, m))$ et $\mathbf{m}_{\mathsf{stop}} \ni \mathtt{s}(\pi, q, m)$, $p, q \in \mathbb{P}$, $m \in \mathbb{V}$.

Démonstration.

Puisque $msc(e')$ est k_{so}-synchrone et $e' = \mathsf{deviate}(e \cdot r)$, $msc(e)$ (sans la dernière réception r) est k_{so}-synchrone.

D'après le Théorème 6.2.7, e' est une exécution mauvaise si et seulement si $\mathsf{GDE}(e \cdot r)$, resp. $\mathsf{GD}(e \cdot r)$ si $\mathsf{com} = \mathsf{pp}$ contient une composante fortement connexe de taille $\geq k+1$. Cette composante fortement connexe doit contenir un sommet associé à la dernière réception r de $e \cdot r$. Dans $\mathsf{GDE}(e')$, resp. $\mathsf{GD}(e')$, cette composante fortement connexe correspond à l'ensemble des sommets qui sont à la fois accessibles depuis $\mathbf{m}_{\mathsf{start}}$ et co-accessibles depuis $\mathbf{m}_{\mathsf{stop}}$.

Puisque les sommets ψ_{start} et $\mathbf{m}_{\mathsf{stop}}$ comptent pour le même sommet dans le graphe de $e \cdot r$, la taille de la composante fortement connexe correspond à la taille de l'ensemble $\mathsf{EPost}^*(\psi_{\mathsf{start}}) \cap \mathsf{EPre}^*(\psi_{\mathsf{stop}})$ moins un, respectivement $\mathsf{Post}^*(\psi_{\mathsf{start}}) \cap \mathsf{Pre}^*(\psi_{\mathsf{stop}})$ moins un.

□

Pour un ensemble de sommets $U \subseteq V$, soit

$$\begin{aligned}
\mathsf{Post}^*(U) &= \bigcup\{\mathsf{Post}^*(\mathbf{m}) \mid \mathbf{m} \in U\},\\
\mathsf{Pre}^*(U) &= \bigcup\{\mathsf{Pre}^*(\mathbf{m}) \mid \mathbf{m} \in U\},\\
\mathsf{EPost}^*(U) &= \bigcup\{\mathsf{EPost}^*(\mathbf{m}) \mid \mathbf{m} \in U\}, \text{ et}\\
\mathsf{EPre}^*(U) &= \bigcup\{\mathsf{EPre}^*(\mathbf{m}) \mid \mathbf{m} \in U\}.
\end{aligned}$$

On définit également ces ensembles pour une séquence d'actions e donnée et un ensemble de processus, pour la vue locale des ensembles des messages accessibles et co-accessibles dans un k_{so}-échange e.

$$\begin{aligned}
\mathsf{Post}_e(P) &= \mathsf{Post}^*(\{\mathbf{m} \mid \mathsf{procs}(\mathbf{m}) \cap P \neq \emptyset\}),\\
\mathsf{Pre}_e(Q) &= \mathsf{Pre}^*(\{\mathbf{m} \mid \mathsf{procs}(\mathbf{m}) \cap Q \neq \emptyset\}),\\
\mathsf{EPost}_e(P) &= \mathsf{EPost}^*(\{\mathbf{m} \mid \mathsf{procs}(\mathbf{m}) \cap P \neq \emptyset\}), \text{ et}\\
\mathsf{EPre}_e(Q) &= \mathsf{EPre}^*(\{\mathbf{m} \mid \mathsf{procs}(\mathbf{m}) \cap Q \neq \emptyset\})
\end{aligned}$$

On définit grâce aux relations de transition suivantes un k_{so}-échange correspondant à une mauvaise exécution en comptant le nombre de messages appartenant à la même composante fortement connexe que le message envoyé à π. On distingue le cas des systèmes communiquant en boîte aux lettres, et des systèmes communiquant en pair à pair. Les relations $\xRightarrow[\mathsf{mb{\cdot}so{\cdot}bad}]{e,k}$ et $\xRightarrow[\mathsf{pp{\cdot}so{\cdot}bad}]{e,k}$ manipulent des configurations abstraites de la forme $(P, Q, \mathtt{cnt})$ où $P, Q \in \mathbb{P}$ et $\mathtt{cnt}$ est un compteur borné à $k + 2$.

$$\frac{\begin{array}{c}P' = \mathsf{procs}(\mathsf{EPost}_e(P)) \quad Q = \mathsf{procs}(\mathsf{EPre}_e(Q'))\\ SCC_e = \mathsf{EPost}_e(P) \cap \mathsf{EPre}_e(Q')\\ \mathtt{cnt}' = \mathsf{min}(k+2, \mathtt{cnt} + n) \text{ où } n = |SCC_e|\end{array}}{(P, Q, \mathtt{cnt}) \xRightarrow[\mathsf{mb{\cdot}so{\cdot}bad}]{e,k} (P', Q', \mathtt{cnt}')}$$

FIGURE 6.39 – Définition de la relation de transition $\xRightarrow[\mathsf{mb{\cdot}so{\cdot}bad}]{e,k}$

$$\frac{\begin{array}{c}P' = \mathsf{procs}(\mathsf{Post}_e(P)) \quad Q = \mathsf{procs}(\mathsf{Pre}_e(Q'))\\ SCC_e = \mathsf{Post}_e(P) \cap \mathsf{Pre}_e(Q')\\ \mathtt{cnt}' = \mathsf{min}(k+2, \mathtt{cnt} + n) \text{ où } n = |SCC_e|\end{array}}{(P, Q, \mathtt{cnt}) \xRightarrow[\mathsf{pp{\cdot}so{\cdot}bad}]{e,k} (P', Q', \mathtt{cnt}')}$$

FIGURE 6.40 – Définition de la relation de transition $\xRightarrow[\mathsf{pp{\cdot}so{\cdot}bad}]{e,k}$

Exemple 6.4.6 – Soit μ le MSC de la Figure 6.41.a. On repère les processus accessibles depuis $\mathbf{m}_{\mathsf{start}} = (s, m_2)$ avec les traits orange et les processus co-accessibles par $\mathbf{m}_{\mathsf{stop}} = m_2$ avec les traits verts. Par exemple, le message m_3 permet au processus q d'être accessible, et le message m_5 permet au processus p d'être co-accessible. Le message m_4 montre le cas particulier d'un message non couplé : il doit être envoyé avant le message m_3 et permet donc lui aussi de transférer la propriété d'accessibilité. Ainsi, les messages m_4 et m_3 associés permettent au processus r d'être co-accessible, et dans l'autre sens, au processus p d'être accessible.

Le graphe de dépendances étendu associé à μ est décrit dans la Figure 6.41.b. On repère les sommets accessibles à leur fond orange et les sommets co-accessibles à leur fond vert : on est alors capables de savoir quels messages appartiennent à la même composante fortement connexe que m_2 si l'on transpose ce MSC dans le système originel. Les sommets sur fond orange et vert sont accessibles et co-accessibles et appartiennent donc à cette composante fortement connexe. On constate alors qu'elle ne contient pas de cycle SS, ce qui signifie que e telle que $msc(e) = \mu$ est faisable. D'autre part, si l'on pose $k < 4$, alors e est mauvaise car cette composante fortement connexe contient 4 messages.

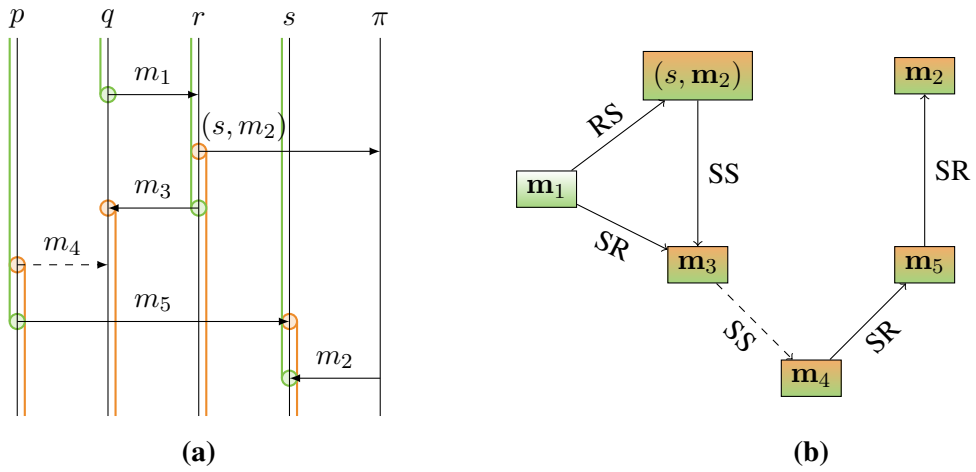


FIGURE 6.41 – Un MSC (a) et son graphe de conflits (a) avec les détails des ensembles $\mathsf{Pre}^*(m_2)$ et $\mathsf{Post}^*((s, m_2))$

Ainsi, grâce à ces relations de transitions, nous sommes capables de reconnaître une exécution mauvaise.

Lemme 6.4.11. *Soit e' une exécution faisable de $\mathcal{S}'$. Alors, e' est mauvaise si et seulement s'il existe $e'' = e_1 \cdots e_n \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$ avec $e_1, \cdots, e_n \in (\mathtt{S}+\mathtt{R})^{\leq 2k}$ et $msc(e') = msc(e'')$, $P', Q \subseteq \mathbb{P}$ tel que*

$$(\{\pi\}, Q, 0) \xrightarrow[\mathsf{mb\cdot so\cdot bad}]{e_1,k} \cdots \xrightarrow[\mathsf{mb\cdot so\cdot bad}]{e_n,k} (P', \{\pi\}, k+2) \textit{ si } \mathsf{com} = \mathsf{mb}$$

$$(\{\pi\}, Q, 0) \xrightarrow[\mathsf{pp\cdot so\cdot bad}]{e_1,k} \cdots \xrightarrow[\mathsf{pp\cdot so\cdot bad}]{e_n,k} (P', \{\pi\}, k+2) \textit{ si } \mathsf{com} = \mathsf{pp}$$

Démonstration.

$\Rightarrow$ Soit $\mathcal{S}$ tel que $\mathsf{com} = \mathsf{mb}$. Soient e' une exécution k-synchronisable faisable et mauvaise et $e'' = e_1 \cdots e_n \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$. Nous montrons, par récurrence sur n, que

$$(\{\pi\}, Q, 0) \xrightarrow[\mathsf{mb\cdot so\cdot bad}]{e_1,k} \cdots \xrightarrow[\mathsf{mb\cdot so\cdot bad}]{e_n,k} (P', \{\pi\}, k+2)$$

Base $n = 1$ Alors $e'' = e_1 \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$. Nous montrons alors que

$$(\{\pi\}, Q, 0) \xRightarrow[\mathsf{mb\cdot so\cdot bad}]{e_1,k} (P', \{\pi\}, k+2)$$

Par le Lemme 6.4.10, on sait que l'ensemble $\mathsf{EPost}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{EPre}^*(\mathbf{m}_{\mathsf{stop}})$ est de taille supérieure ou égale à $k+2$, ce qui conclut le cas de base.

Récurrence $e'' = e_1 \cdots e_{n-1} \cdot e_n \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$ tel que

$$(\{\pi\}, Q, 0) \xRightarrow[\mathsf{mb\cdot so\cdot bad}]{e_1,k} \cdots \xRightarrow[\mathsf{mb\cdot so\cdot bad}]{e_{n-1},k} (P', Q', \mathtt{cnt})$$

Montrons que $(P', Q', \mathtt{cnt}) \xRightarrow[\mathsf{mb\cdot so\cdot bad}]{e_n,k} (P'', \{\pi\}, \mathtt{cnt}')$.

Supposons que la taille de l'ensemble $\mathsf{EPost}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{EPre}^*(\mathbf{m}_{\mathsf{stop}})$ est supérieur ou égale à $k+2$. Nous montrons alors que tous les sommets de $\mathsf{EPost}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{EPre}^*(\mathbf{m}_{\mathsf{stop}})$ ont été comptés soit dans $e_1 \cdots e_{n-1}$ soit dans le dernier k_{so}-échange. Prenons $\mathbf{m} \in \mathsf{EPost}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{EPre}^*(\mathbf{m}_{\mathsf{stop}})$ alors il existe un chemin $\mathbf{m}_{\mathsf{start}} \to^* v \to^* \mathbf{m}_{\mathsf{stop}}$ et $\mathbf{m}$ est un message qui appartient soit à $e_1 \cdots e_{n-1}$ soit au dernier k_{so}-échange.

- Si $\mathbf{m}$ appartient à $e_1 \cdots e_{n-1}$, alors on peut diviser le chemin en deux parties telles que $\mathbf{m}_{\mathsf{start}} \to^* v \to^* \mathbf{m}_1$ est dans $\mathsf{GDE}(e_1 \cdots e_{n-1})$, $\mathbf{m}_2 \to^* \mathbf{m}_{\mathsf{stop}}$ est dans $\mathsf{GDE}(e_n)$ et $\mathsf{procs}(\mathbf{m}_1) \cup \mathsf{procs}(\mathbf{m}_2) = \{p\} \neq \emptyset$. À partir de là, on peut déduire que le processus $p \in Q'$ et donc $\mathbf{m} \in \mathsf{EPre}_{e_1 \cdots e_{n-1}}(Q')$. De plus, $\mathbf{m} \in \mathsf{EPost}_{e_1 \cdots e_{n-1}}(\pi)$ et donc le sommet $\mathbf{m}$ est compté dans $e_1 \cdots e_{n-1}$.
- De la même façon, si $\mathbf{m}$ appartient au dernier k_{so}-échange, on peut diviser le chemin en deux parties telles que $\mathbf{m}_{\mathsf{start}} \to^* \mathbf{m}_1$ est dans $\mathsf{GDE}(e_1 \cdots e_{n-1})$, $\mathbf{m}_2 \to^* \mathbf{m} \to^* \mathbf{m}_{\mathsf{stop}}$ est dans $\mathsf{GDE}(e_n)$ et $\mathsf{procs}(\mathbf{m}_1) \cup \mathsf{procs}(\mathbf{m}_2) = \{p\} \neq \emptyset$. À partir de là, on peut déduire que $p \in P'$ et donc $\mathbf{m} \in \mathsf{EPost}_{e_n}(P')$. De plus, $\mathbf{m} \in \mathsf{EPre}_{e_n}(\pi)$ et donc le sommet $\mathbf{m}$ est compté dans le dernier k_{so}-échange.

Ainsi, tous les sommets de $\mathsf{EPost}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{EPre}^*(\mathbf{m}_{\mathsf{stop}})$ sont considérés et si $\mathsf{EPost}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{EPre}^*(\mathbf{m}_{\mathsf{stop}}) \geq k+2$ et donc la variable $\mathtt{cnt}' \geq k+2$, ce qui conclue cette partie de la preuve.

On peut facilement adapter cette preuve aux systèmes communiquant en pair à pair en changeant les graphes de dépendances étendus par les graphes de dépendances, et de même pour les ensembles de sommets accessibles et co-accessibles.

$\Leftarrow$ Soit e' une exécution k_{so}-synchronisable et faisable, $e'' = e_1 \cdots e_n \cdot \mathtt{s}(\pi, q, m) \cdot \mathtt{r}(\pi, q, m)$ avec $msc(e') = msc(e'')$, and $P', Q \subseteq \mathbb{P}$, $\mathtt{cnt} \in \{0, \ldots, k+2\}$ tels que

$$(\{\pi\}, Q, 0) \xRightarrow[\mathsf{mb\cdot so\cdot bad}]{e_1,k} \ldots \xRightarrow[\mathsf{mb\cdot so\cdot bad}]{e_n,k} (P', \{\pi\}, \mathtt{cnt})$$

On suppose que $\mathtt{cnt} = k+2$. e' est faisable par le Lemme 6.4.9. Chaque sommet $\mathbf{m}$ appartient à $\mathsf{EPre}_{e_i}(P_i) \cap \mathsf{EPost}_{e_i}(Q'_i) \setminus \mathbf{m}_{\mathsf{start}}$ mais également à $\mathsf{EPost}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{EPre}^*(\mathbf{m}_{\mathsf{stop}})$ alors $\mid \mathsf{EPost}^*(\mathbf{m}_{\mathsf{start}}) \cap \mathsf{EPre}^*(\mathbf{m}_{\mathsf{stop}}) \mid \geq k+2$. Alors, e' est mauvaise, ce qui est conclut cette preuve pour la communication en boîte aux lettres.

On peut facilement adapter cette preuve à la communication en pair à pair, utilisant la transition $\xrightarrow[\mathsf{pp{\cdot}so{\cdot}bad}]{e,k}$.

□

Enfin, nous pouvons conclure par le théorème suivant constituant le résultat final de cette section.

Théorème 6.4.12. *La k_{so}-synchronisabilité pour un système $\mathcal{S}$ et un k donné est décidable.*

Démonstration.

Soit $\mathcal{S}$ un système donné. Par les Lemmes 6.4.7, 6.4.9 et 6.4.11, $\mathcal{S}$ n'est pas k_{so}-synchronisable si et seulement s'il existe une séquence d'action $e' = e'_1 \cdots e'_n \cdot s \cdot r$ tel que $e_i \in \mathtt{S}^{\leq k}\mathtt{R}^{\leq k}$, $s = \mathtt{s}(\pi, q, m)$, $r = \mathtt{r}(\pi, q, m)$,

$$(\vec{\ell_0}, \overrightarrow{bu_0}, \bot) \xrightarrow[\mathsf{so{\cdot}feas}]{e'_1,k} \cdots \xrightarrow[\mathsf{so{\cdot}feas}]{e'_n,k} (\vec{\ell}, \vec{bu}, \mathsf{c_{dest}})$$

et, si $\mathsf{com} = \mathsf{mb}$

$$(\{\pi\}, Q, 0) \xrightarrow[\mathsf{mb{\cdot}so{\cdot}bad}]{e'_1,k} \cdots \xrightarrow[\mathsf{mb{\cdot}so{\cdot}bad}]{e'_n,k} \xrightarrow[\mathsf{mb{\cdot}so{\cdot}bad}]{s\cdot r,k} (P', \{\pi\}, \mathtt{cnt})$$

ou, si $\mathsf{com} = \mathsf{pp}$

$$(\{\pi\}, Q, 0) \xrightarrow[\mathsf{pp{\cdot}so{\cdot}bad}]{e'_1,k} \cdots \xrightarrow[\mathsf{pp{\cdot}so{\cdot}bad}]{e'_n,k} \xrightarrow[\mathsf{pp{\cdot}so{\cdot}bad}]{s\cdot r,k} (P', \{\pi\}, \mathtt{cnt})$$

pour certains $\vec{\ell}, \vec{bu}, Q, P', \mathsf{c_{dest}}$. Puisque $\xrightarrow[\mathsf{so{\cdot}feas}]{e,k}$ et $\xrightarrow[\mathsf{mb{\cdot}so{\cdot}bad}]{e,k}$, et respectivement $\xrightarrow[\mathsf{pp{\cdot}so{\cdot}bad}]{e,k}$, sont des relations sur des ensembles finis, l'existence d'une telle séquence d'actions est décidable.

□

6.5 Conclusion

Dans ce chapitre, nous nous sommes interrogés sur certaines conséquences contre-intuitives de la définition de la k-synchronisabilité.

Nous en avons déduit une nouvelle contrainte qui nous a permis de définir une nouvelle classe : la division en k-échanges doit correspondre à une exécution du système. Cette nouvelle classe a ensuite été élargie en relâchant la contrainte concernant l'ordre entre envois et réceptions au sein d'un k-échange. Les k-échanges deviennent alors des k_{so}-échanges représentant de simples barrières de synchronisation.

Nous nous sommes ensuite attardés sur une comparaison précise de toutes ces classes de systèmes, et ce dans les deux types de communication, avant d'apporter des preuves alternatives à la décidabilité des problèmes d'accessibilité et d'appartenance pour les deux nouvelles classes.

CHAPITRE 7

Conclusion et Perspectives

7.1 Conclusion

La k-synchronisabilité fut le cœur de cette thèse. Les travaux de [Bouajjani et al., 2018a] nous ont poussés à étudier cette propriété en profondeur ainsi que les notions inévitablement liées.

Tout d'abord, les problèmes de vérification tels que l'accessibilité d'un état de contrôle et les approximations, autre que la k-synchronisabilité, permettant la décidabilité de ces problèmes furent répertoriés dans le Chapitre 3. Nous avions précédemment inventorié dans le Chapitre 2 les différentes relations causales pour enfin choisir et définir la réalisabilité, proche de la livraison causale utilisée dans les travaux initiaux de [Bouajjani et al., 2018a], mais plus adaptée à l'étude de la k-synchronisabilité.

Grâce à ce changement, nous avons apporté dans le Chapitre 4 une preuve corrigée de l'accessibilité d'un état de contrôle dans un système k-synchronisable. Cette preuve utilise deux caractérisations graphiques : celle d'un MSC réalisable et celle d'un MSC k-synchrone. En effet, le graphe de dépendances étendu associé à un MSC permet d'identifier s'il s'agit d'un MSC réalisable, et chaque composante fortement connexe dans le graphe de dépendances regroupe les messages appartenant au même k-échange. Alors, grâce à ces caractérisations, l'ensemble des MSC réalisables et k-synchrones peut être représenté par un automate fini où les états contiennent les états de contrôle du système. Ainsi, l'accessibilité y est décidable.

Ce résultat nous a permis, dans la suite du Chapitre 4, d'étudier le problème suivant : pour un k donné, un système donné est-il k-synchronisable ? Bien qu'inspirée de la démarche de la preuve existante dans [Bouajjani et al., 2018a], la preuve fut retravaillée dans les détails avec les modifications dues à la réalisabilité et la correction apportée à la caractérisation graphique d'un MSC k-synchrone.

Si le k est donné en paramètre, nous sommes alors capables de savoir si un système est k-synchronisable. Le problème suivant, lui, était ouvert jusqu'à présent : pour un système donné, existe-t-il un k tel que le système est k-synchronisable ? Nous y répondons dans le Chapitre 5. La trame de la preuve consiste en la recherche du plus grand k-échange premier qui appartient à une exécution du système. Cette recherche se décompose alors en la construction de tous les k-échanges possibles du système puis réduit cet ensemble à ceux appartenant à une exécution. Les constructions nécessaires à ces étapes s'avèrent être des automates finis, ce qui nous permet alors de trouver la taille du plus long mot contenu dans le langage résultant.

Enfin, l'étude approfondie de la k-synchronisabilité a nourri quelques critiques qui peuvent être faites à cette définition : certains cas contre-intuitifs sont inclus dans l'ensemble des sys-

tèmes k-synchronisables. Ces critiques sont explicitées dans le Chapitre 6 et sont suivies de propositions de nouvelles classes de systèmes, les systèmes k_F-synchronisables et les systèmes k_{so}-synchronisables, constituant des variations de la k-synchronisabilité. La suite du chapitre compare les classes nouvellement définies à la k-synchronisabilité ainsi qu'aux systèmes k-bornés. De nombreux exemples sont alors fournis. Le chapitre se termine, par un souci de complétude, par les preuves de décidabilité de l'accessibilité d'un état de contrôle pour les nouvelles classes, ainsi que celui de l'appartenance avec le k donné en paramètre pour la k_{so}-synchronisabilité et sans k en paramètre pour la k_F-synchronisabilité.

Pour parfaire la comparaison entre les systèmes, nous pouvons ajouter un tableau récapitulatif de la décidabilité des problèmes d'appartenance, en considérant que le paramètre k est fourni dans un premier temps, et sans aucun paramètre fourni dans un second, et ce pour un système quelconque et pour chacune des classes vues dans cette thèse. Ce tableau est disponible Figure 7.1, les résultats sont accompagnés des articles dans lesquels ils ont été publiés lorsque ceux-ci existent. Enfin, les résultats en gras indiquent que les preuves correspondantes sont présentes dans cette thèse.

	Avec k	Sans k
$\exists$-k-pp-borné	Décidable [Genest et al., 2007]	Indécidable [Genest et al., 2007]
$\exists$-k-mb-borné	Décidable [Bollig et al., 2021]	-
$\forall$-k-pp-borné	Décidable [Genest et al., 2007]	Indécidable [Genest et al., 2007]
$\forall$-k-mb-borné	Conjecturé décidable [Bollig et al., 2021]	-
k-synchronisabilité en pp	**Décidable** [Di Giusto et al., 2020]	-
k-synchronisabilité en mb	**Décidable** [Di Giusto et al., 2020]	**Décidable** [Di Giusto et al., 2021b]
k_F-synchronisabilité en pp	**Décidable**	-
k_F-synchronisabilité en mb	**Décidable** [Bollig et al., 2021]	**Décidable**
k_{so}-synchronisabilité en pp	**Décidable**	-
k_{so}-synchronisabilité en mb	**Décidable**	-

FIGURE 7.1 – Tableau récapitulatif des résultats de décidabilité pour les problèmes d'appartenance

7.2 Perspectives

Les récents travaux de [Bollig et al., 2021] permettent de généraliser la décidabilité de l'accessibilité et de l'appartenance avec un paramètre k pour de nombreuses classes de systèmes communicants avec une méthode générique. Cela ouvre alors l'imagination à la définition de nouvelles classes de systèmes, pour lesquelles ces problèmes seraient décidables.

Le problème de l'appartenance sans avoir de paramètre k ne peut cependant pas être résolu avec cette méthode. En particulier, il reste encore ouvert notamment pour les systèmes k_{so}-synchronisables. Mais il serait pertinent de trouver une autre technique générique qui serait, elle, capable de répondre à ce problème. Rappelons que ce problème est d'ailleurs indécidable pour la borne existentielle. Alors, reste la question de trouver la classe idéale, englobant un maximum de systèmes communicants tout en admettant la décidabilité de tous ces problèmes.

Un autre domaine pourrait être le sujet d'études complémentaires, celui des différentes classes causales. Il serait pertinent de pouvoir catégoriser et caractériser la causalité qui peut prendre des formes variables selon la définition choisie.

Bibliographie

[Abdulla et al., 1996] Abdulla, P., Cerans, K., Jonsson, B., and Yih-Kuen Tsay (1996). General decidability theorems for infinite-state systems. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 313–321, New Brunswick, NJ, USA. IEEE Comput. Soc. Press.

[Abdulla et al., 2012] Abdulla, P. A., Atig, M. F., and Cederberg, J. (2012). Timed Lossy Channel Systems. In *FSTTCS*.

[Abdulla and Jonsson, 1996] Abdulla, P. A. and Jonsson, B. (1996). Verifying Programs with Unreliable Channels. *Information and Computation*, 127(2) :91–101.

[Akroun and Salaün, 2018] Akroun, L. and Salaün, G. (2018). Automated verification of automata communicating via FIFO and bag buffers. *Formal Methods in System Design*, 52(3) :260–276.

[Alur and Dill, 1994] Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, 126(2) :183–235.

[Alur et al., 2003] Alur, R., Etessami, K., and Yannakakis, M. (2003). Inference of Message Sequence Charts. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, page 15.

[Alur et al., 1996] Alur, R., Holzmann, G. J., and Peled, D. (1996). An analyzer for message sequence charts. In Margaria, T. and Steffen, B., editors, *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, pages 35–48, Berlin, Heidelberg. Springer.

[Basu and Bultan, 2016] Basu, S. and Bultan, T. (2016). On deciding synchronizability for asynchronously communicating systems. *Theoretical Computer Science*, 656 :60–75.

[Bengtsson et al., 1996] Bengtsson, J., Larsen, K., Larsson, F., Pettersson, P., and Yi, W. (1996). UPPAAL — a tool suite for automatic verification of real-time systems. In Alur, R., Henzinger, T. A., and Sontag, E. D., editors, *Hybrid Systems III*, Lecture Notes in Computer Science, pages 232–243, Berlin, Heidelberg. Springer.

[Birman and Joseph, 1987] Birman, K. P. and Joseph, T. A. (1987). Reliable communication in the presence of failures. *ACM Transactions on Computer Systems*, 5(1) :47–76.

[Boigelot and Godefroid, 1999] Boigelot, B. and Godefroid, P. (1999). Symbolic Verification of Communication Protocols with Infinite State Spaces using QDDs. *Formal Methods in System Design*, 14(3) :237–255.

[Boigelot et al., 1997] Boigelot, B., Godefroid, P., Willems, B., and Wolper, P. (1997). The power of QDDs (extended abstract). In Van Hentenryck, P., editor, *Static Analysis*, Lecture Notes in Computer Science, pages 172–186, Berlin, Heidelberg. Springer.

[Bollig et al., 2021] Bollig, B., Di Giusto, C., Finkel, A., Laversa, L., Lozes, E., and Suresh, A. (2021). A unifying framework for deciding synchronizability. In *32nd International Conference on Concurrency Theory (CONCUR 2021)*.

[Bollig et al., 2020] Bollig, B., Finkel, A., and Suresh, A. (2020). Bounded Reachability Problems are Decidable in FIFO Machines. In *31st International Conference on Concurrency Theory (CONCUR 2020)*, Proceedings of the 31st International Conference on Concurrency Theory (CONCUR 2020), Vienna, Austria.

[Bollig et al., 2018] Bollig, B., Fortin, M., and Gastin, P. (2018). It is easy to be wise after the event : Communicating finite-state machines capture first-order logic with" happened before". In *29th International Conference on Concurrency Theory (CONCUR 2018)*, volume 118, pages 7–1. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[Bollig et al., 2014] Bollig, B., Gastin, P., and Schubert, J. (2014). Parameterized Verification of Communicating Automata under Context Bounds. In Ouaknine, J., Potapov, I., and Worrell, J., editors, *Reachability Problems*, Lecture Notes in Computer Science, pages 45–57, Cham. Springer International Publishing.

[Bouajjani and Emmi, 2014] Bouajjani, A. and Emmi, M. (2014). Bounded phase analysis of message-passing programs. *International Journal on Software Tools for Technology Transfer*, 16(2) :127–146.

[Bouajjani et al., 2018a] Bouajjani, A., Enea, C., Ji, K., and Qadeer, S. (2018a). On the Completeness of Verifying Message Passing Programs Under Bounded Asynchrony. In Chockler, H. and Weissenbacher, G., editors, *Computer Aided Verification*, Lecture Notes in Computer Science, pages 372–391. Springer International Publishing.

[Bouajjani et al., 2018b] Bouajjani, A., Enea, C., Ji, K., and Qadeer, S. (2018b). On the Completeness of Verifying Message Passing Programs under Bounded Asynchrony. *arXiv :1804.06612 [cs]*. arXiv : 1804.06612.

[Bouyer et al., 2012] Bouyer, P., Markey, N., Ouaknine, J., Schnoebelen, P., and Worrell, J. (2012). On termination and invariance for faulty channel machines. *Formal Aspects of Computing*, 24(4) :595–607.

[Brand and Zafiropulo, 1983] Brand, D. and Zafiropulo, P. (1983). On Communicating Finite-State Machines. *Journal of the ACM*, 30(2) :323–342.

[Cécé and Finkel, 2005] Cécé, G. and Finkel, A. (2005). Verification of programs with half-duplex communication. *Information and Computation*, 202(2) :166–190.

[Cécé et al., 1996] Cécé, G., Finkel, A., and Purushothaman Iyer, S. (1996). Unreliable Channels Are Easier to Verify Than Perfect Channels. *Information and Computation*, 124(1) :20–31.

[CCITT, 1992] CCITT, S. (1992). Draft recommendation Z. 120 : Message sequence chart. *Submitted to CCITT*.

[Chaouch-Saad et al., 2009] Chaouch-Saad, M., Charron-Bost, B., and Merz, S. (2009). A Reduction Theorem for the Verification of Round-Based Distributed Algorithms. In Bournez,

O. and Potapov, I., editors, *Reachability Problems*, Lecture Notes in Computer Science, pages 93–106, Berlin, Heidelberg. Springer.

[Charron-Bost et al., 1996] Charron-Bost, B., Mattern, F., and Tel, G. (1996). Synchronous, asynchronous, and causally ordered communication. *Distributed Computing*, 9(4) :173–191.

[Charron-Bost and Schiper, 2009] Charron-Bost, B. and Schiper, A. (2009). The Heard-Of model : computing in distributed systems with benign faults. *Distributed Computing*, 22(1) :49–71.

[Chevrou et al., 2016] Chevrou, F., Hurault, A., and Quéinnec, P. (2016). On the diversity of asynchronous communication. *Formal Aspects of Computing*, 28(5) :847–879.

[Choquet and Finkel, 1987] Choquet, A. and Finkel, A. (1987). *Simulation of Linear FIFO Nets by a new Class of Petri Nets*. Université de Paris-Sud. Centre d'Orsay. Laboratoire de Recherche en ….

[Clemente et al., 2014] Clemente, L., Herbreteau, F., and Sutre, G. (2014). Decidable Topologies for Communicating Automata with FIFO and Bag Channels. In *CONCUR 2014 - 25th International Conference on Concurrency Theory*, volume 8704 of *LNCS*, pages 281–296, Rome, Italy. Springer.

[Cousot, 2019] Cousot, P. (2019). Calculational Design of a Regular Model Checker by Abstract Interpretation : 16th International Colloquium on Theoretical Aspects of Computing, ICTAC 2019. *Theoretical Aspects of Computing – ICTAC 2019 - 16th International Colloquium, Proceedings*, pages 3–21. Publisher : Springer.

[Damm and Harel, 2001] Damm, W. and Harel, D. (2001). LSCs : Breathing Life into Message Sequence Charts. *Formal Methods in System Design*, 19(1) :45–80.

[David and Alla, 1994] David, R. and Alla, H. (1994). Petri nets for modeling of dynamic systems : A survey. *Automatica*, 30(2) :175–202.

[Di Giusto et al., 2021a] Di Giusto, C., Germerie-Guizouarn, L., and Lozes, E. (2021a). Towards Generalised Half-Duplex Systems. *International Conference on Engineering, Technology and Innovation*.

[Di Giusto et al., 2020] Di Giusto, C., Laversa, L., and Lozes, E. (2020). On the k-synchronizability of systems. In *23rd International Conference on Foundations of Software Science and Computer Systems (FOSSACS 2020)*, volume 12077, pages 157–176. Springer.

[Di Giusto et al., 2021b] Di Giusto, C., Laversa, L., and Lozes, E. (2021b). Guessing the Buffer Bound for k-Synchronizability. In Maneth, S., editor, *Implementation and Application of Automata*, Lecture Notes in Computer Science, pages 102–114, Cham. Springer International Publishing.

[Drăgoi et al., 2016] Drăgoi, C., Henzinger, T. A., and Zufferey, D. (2016). PSync : a partially synchronous language for fault-tolerant distributed algorithms. *ACM SIGPLAN Notices*, 51(1) :400–415.

[Elrad and Francez, 1982] Elrad, T. and Francez, N. (1982). Decomposition of distributed programs into communication-closed layers. *Science of Computer Programming*, 2(3) :155–173.

[Esparza et al., 2012] Esparza, J., Ganty, P., and Majumdar, R. (2012). A perfect model for bounded verification. In *2012 27th Annual IEEE Symposium on Logic in Computer Science*, pages 285–294. IEEE.

[Fernandez et al., 1996] Fernandez, J. C., Garavel, H., Kerbrat, A., Mounier, L., Mateescu, R., and Sighireanu, M. (1996). CADP a protocol validation and verification toolbox. In Goos, G., Hartmanis, J., Leeuwen, J., Alur, R., and Henzinger, T. A., editors, *Computer Aided Verification*, volume 1102, pages 437–440. Springer Berlin Heidelberg, Berlin, Heidelberg. Series Title : Lecture Notes in Computer Science.

[Finkel, 1990] Finkel, A. (1990). Reduction and covering of infinite reachability trees. *Information and Computation*, 89(2) :144–179.

[Finkel and Lozes, 2017] Finkel, A. and Lozes, E. (2017). Synchronizability of Communicating Finite State Machines is not Decidable. In Chatzigiannakis, I., Indyk, P., Kuhn, F., and Muscholl, A., editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 122 :1–122 :14, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISSN : 1868-8969.

[Finkel and Praveen, 2019] Finkel, A. and Praveen, M. (2019). Verification of Flat FIFO Systems. In *CONCUR 2019*, AMSTERDAM, Netherlands.

[Finkel and Schnoebelen, 2001] Finkel, A. and Schnoebelen, P. (2001). Well-structured transition systems everywhere ! *Theoretical Computer Science*, 256(1) :63–92.

[Francalanza et al., 2018] Francalanza, A., Pérez, J. A., and Sánchez, C. (2018). Runtime Verification for Decentralised and Distributed Systems. In Bartocci, E. and Falcone, Y., editors, *Lectures on Runtime Verification : Introductory and Advanced Topics*, Lecture Notes in Computer Science, pages 176–210. Springer International Publishing, Cham.

[Genest et al., 2006] Genest, B., Kuske, D., and Muscholl, A. (2006). A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Information and Computation*, 204(6) :920–956.

[Genest et al., 2007] Genest, B., Kuske, D., and Muscholl, A. (2007). On communicating automata with bounded channels. *Fundamenta Informaticae*, 80(1-3) :147–167.

[Ghosh, 2014] Ghosh, S. (2014). *Distributed Systems : An Algorithmic Approach, Second Edition*. CRC Press. Google-Books-ID : 60fSBQAAQBAJ.

[Gouda et al., 1987] Gouda, M. G., Gurari, E. M., Lai, T.-H., and Rosier, L. E. (1987). On deadlock detection in systems of communicating finite state machines. *On deadlock detection in systems of communicating finite state machines*, 6(3) :209–228. Place : Bratislava Publisher : Slovak Academy of Sciences.

[Grabowski et al., 1993] Grabowski, J., Graubmann, P., and Rudolph, E. (1993). The standardization of message sequence charts. In *Proceedings 1993 Software Engineering Standards Symposium*, pages 48–63.

[Harel and Thiagarajan, 2003] Harel, D. and Thiagarajan, P. S. (2003). Message Sequence Charts. In Lavagno, L., Martin, G., and Selic, B., editors, *UML for Real : Design of Embedded Real-Time Systems*, pages 77–105. Springer US, Boston, MA.

[Heussner et al., 2012] Heussner, A., Leroux, J., Muscholl, A., and Sutre, G. (2012). Reachability Analysis of Communicating Pushdown Systems. *Logical Methods in Computer Science*, 8(3). arXiv : 1209.0359.

[Hoare, 1978] Hoare, C. A. R. (1978). Communicating sequential processes. *Communications of the ACM*, 21(8) :666–677.

[Holzmann and Peled, 1996] Holzmann, G. J. and Peled, D. (1996). The state of Spin. In Alur, R. and Henzinger, T. A., editors, *Computer Aided Verification*, Lecture Notes in Computer Science, pages 383–389, Berlin, Heidelberg. Springer.

[Jéron and Jard, 1993] Jéron, T. and Jard, C. (1993). Testing for unboundedness of fifo channels. *Theoretical Computer Science*, 113(1) :93–117.

[Kragl et al., 2018] Kragl, B., Qadeer, S., and Henzinger, T. A. (2018). Synchronizing the asynchronous. In *29th International Conference on Concurrency Theory*.

[Kuske and Muscholl, 2010] Kuske, D. and Muscholl, A. (2010). Communicating automata. -.

[La Torre et al., 2007] La Torre, S., Madhusudan, P., and Parlato, G. (2007). A Robust Class of Context-Sensitive Languages. In *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 161–170. ISSN : 1043-6871.

[La Torre et al., 2008] La Torre, S., Madhusudan, P., and Parlato, G. (2008). Context-Bounded Analysis of Concurrent Queue Systems. In Ramakrishnan, C. R. and Rehof, J., editors, *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, pages 299–314. Springer Berlin Heidelberg.

[Lamport, 1978] Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications*.

[Lange et al., 2015] Lange, J., Tuosto, E., and Yoshida, N. (2015). From Communicating Machines to Graphical Choreographies. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages - POPL '15*, pages 221–232, Mumbai, India. ACM Press.

[Lange and Yoshida, 2019] Lange, J. and Yoshida, N. (2019). Verifying asynchronous interactions via communicating session automata. In Dillig, I. and Tasiran, S., editors, *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I*, volume 11561 of *Lecture Notes in Computer Science*, pages 97–117. Springer.

[Lipton, 1975] Lipton, R. J. (1975). Reduction : a method of proving properties of parallel programs. *Communications of the ACM*, 18(12) :717–721.

[Mattern and Fünfrocken, 1995] Mattern, F. and Fünfrocken, S. (1995). A non-blocking lightweight implementation of causal order message delivery. In Goos, G., Hartmanis, J., Leeuwen, J., Birman, K. P., Mattern, F., and Schiper, A., editors, *Theory and Practice in Distributed Systems*, volume 938, pages 197–213. Springer Berlin Heidelberg, Berlin, Heidelberg. Series Title : Lecture Notes in Computer Science.

[Milner, 1980] Milner, R. (1980). *A Calculus of Communicating Systems*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg.

[Milner, 1999] Milner, R. (1999). *Communicating and Mobile Systems : The Pi Calculus*. Cambridge University Press. Google-Books-ID : ex6Xkj50ULkC.

[Muscholl and Peled, 1999] Muscholl, A. and Peled, D. (1999). Message Sequence Graphs and Decision Problems on Mazurkiewicz Traces. In Kutyłowski, M., Pacholski, L., and Wierzbicki, T., editors, *Mathematical Foundations of Computer Science 1999*, Lecture Notes in Computer Science, pages 81–91, Berlin, Heidelberg. Springer.

[Muscholl et al., 1998] Muscholl, A., Peled, D., and Su, Z. (1998). Deciding properties for message sequence charts. In Goos, G., Hartmanis, J., van Leeuwen, J., and Nivat, M., editors, *Foundations of Software Science and Computation Structures*, volume 1378, pages 226–242. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Ouaknine and Worrell, 2005] Ouaknine, J. and Worrell, J. (2005). On the decidability of metric temporal logic. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*, pages 188–197. ISSN : 1043-6871.

[Pachl, 2012] Pachl, J. (2012). Reachability problems for communicating finite state machines. *arXiv :cs/0306121*. arXiv : cs/0306121.

[Papadimitriou, 1979] Papadimitriou, C. H. (1979). The serializability of concurrent database updates. *Journal of the ACM*, 26(4) :631–653.

[Recommendation, 2011] Recommendation, I. (2011). Z. 120 : Message sequence chart. Technical report, Technical report, ITU-T, Geneva.

[Sistla, 1994] Sistla, A. P. (1994). Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6(5) :495–511.

[Tanenbaum and van Steen, 2016] Tanenbaum, A. and van Steen, M. (2016). *Distributed Systems 3/e*. Amazon.

[v. Gleissenthall et al., 2019] v. Gleissenthall, K., Kıcı, R. G., Bakst, A., Stefan, D., and Jhala, R. (2019). Pretend synchrony : synchronous verification of asynchronous distributed programs. *Proceedings of the ACM on Programming Languages*, 3(POPL) :1–30.

[Wolfgang et al., 1997] Wolfgang, T., Rozenberg, G., and Salomaa, A. (1997). Languages, automata, and logic. *Handbook of formal languages*, pages 389–455.

[Yu et al., 1999] Yu, Y., Manolios, P., and Lamport, L. (1999). Model Checking TLA+ Specifications. In Pierre, L. and Kropf, T., editors, *Correct Hardware Design and Verification Methods*, Lecture Notes in Computer Science, pages 54–66, Berlin, Heidelberg. Springer.

Liste des figures

Liste des définitions

Liste des exemples

La synchronisabilité pour les systèmes distribués

Laetitia LAVERSA

Résumé

Les systèmes distribués sont omniprésents, cependant leur implémentation est complexe et sujette aux erreurs. Pour les vérifier, ils peuvent être modélisés en système d'automates communicants, où chaque automate représente le comportement d'un élément du système. Les propriétés liées à la vérification, telles que le problème de l'accessibilité, restent indécidables dans un tel modèle. En effet, un système d'automates communiquant de façon asynchrone est Turing équivalent. C'est pourquoi l'utilisation d'approximations est donc nécessaire. La k-synchronisabilité est l'une de ces approximations. Un système est k-synchronisable si pour toute exécution, il existe une exécution équivalente pouvant être divisée en phases, appelées k-échanges, contenant k messages. Dans cette thèse, nous nous intéressons à l'analyse des systèmes k-synchronisables : nous montrons que le problème de l'accessibilité est décidable et nous nous interrogeons sur la k-synchronisabilité d'un système donné. Pour ce deuxième point, nous regardons le cas où un k est passé en paramètre, et le cas où il faut trouver le k tel que le système est k-synchronisable. Nous prouvons que ces deux problèmes sont décidables lorsque le système communique en boîte aux lettres. Le premier problème est également décidable en pair-à-pair, mais, avec ce type de communication, le deuxième problème reste ouvert. Nous complétons cette étude en identifiant certaines implications contre-intuitives de la notion de k-synchronisabilité, celles-ci nous ont poussés à définir des variantes de la k-synchronisabilité. Une étude comparative de différentes classes de systèmes, soit étudiées ou définies dans cette thèse, soit provenant de la littérature, conclut nos travaux.

Mots-clés : Automates communicants, Synchronisabilité, Diagrammes de séquences

Abstract

Distributed systems are ubiquitous and their implementation is complex and error-prone. In order to check for errors, they can be modeled as systems of communicating automata, where each automaton represents the behavior of an element of the system. Verification problems such as reachability are undecidable in such a model. Indeed, a system of communicating automata is Turing-equivalent. For that, the use of approximations is necessary. k-synchronizability is one of these techniques. A system is k-synchronizable if, for all execution, there is an equivalent execution that can be divided into phases containing k messages. These phases are called k-exchanges. In this thesis, we analyse k-synchronizable systems: we show that reachability is decidable and we are interested in the membership problem, that is: given a system, decide whether it is k-synchronizable. We study both the case where k is an input of the problem, and the case where we have to guess a k such that the system is k-synchronizable. We study them according to the type of communication of the system. Either the system is in mailbox and so each process has only one buffer to store all received messages, or the system is in peer-to-peer, and each process has a buffer for each sender. Both problems are decidable when the system is communicating in mailbox. When the system is communicating in peer-to-peer, the first one is decidable and the second remains open. Finally, we point out some counter-intuitive cases of k-synchronizable systems that lead us to propose some variations to the definition of k-synchronizability. A comparative study of the state-of-art classes of systems and our new classes concludes our work.

Keywords: Communicating automata, Synchronizability, Message sequence charts