ENSC-2006 n° 6

THÈSE

présentée à l'École Normale Supérieure de Cachan

pour obtenir le grade de

Docteur de l'École Normale Supérieure de Cachan

par : Nathalie BERTRAND

Spécialité : INFORMATIQUE

Modèles stochastiques pour les pertes de messages dans les protocoles asynchrones, et techniques de vérification automatique.

Soutenue le 6 octobre 2006, devant un jury composé de :

- Christel BAIER — examinatrice
- Danièle BEAUQUIER — présidente du jury
- Ahmed BOUAJJANI — examinateur
- Jean-François RASKIN — rapporteur
- Philippe SCHNOEBELEN — directeur de thèse
- Igor WALUKIEWICZ — rapporteur

Résumé

Les protocoles de communication asynchrones sont naturellement modélisés par des automates communicants via des canaux FIFO non-bornés. Dans cette thèse nous nous intéressons aux variantes des LCS (Lossy Channel Systems) pour lesquelles les pertes de messages dans les canaux se font de façon probabiliste. Plus précisément, on considère tour à tour des sémantiques sous forme de chaînes de Markov et de processus de décision markoviens. Grâce à un théorème général de convergence de certains points fixes dans les systèmes de transition bien structurés, nous prouvons pour les deux modèles (PLCS et NPLCS) de nombreux résultats de décidabilité vis-à-vis de la vérification de propriétés du temps linéaire. Nous donnons également les limites des modèles par l'intermédiaire de résultats d'indécidabilité. Un prototype a fait l'objet de l'implémentation des algorithmes développés dans la thèse. Malgré la grande complexité des problèmes (non primitifs récursifs) cet outil a permis de prouver des propriétés de vivacité sur des protocoles tels que le Bit Alterné et le protocole de Pachl.

Mots-clefs : Vérification formelle, Lossy Channel Systems, Probabilités.

Abstract

Asynchronous communication protocols are naturally seen as communicating finite automata over unbounded FIFO channels. In this thesis, we consider variants of LCS (Lossy Channel Systems) for which message losses are probabilistic. More precisely, we introduce the models of Probabilistic LCS (PLCS), and Nondeterministic and Probabilistic LCS (NPLCS) whose semantics are respectively Markov chains and Markov decision processes. A general criterion on convergence of fixed points expressions in Well Structured Transition Systems allows to prove a number of decidability results with respect to verification of linear-time properties for both models. We also prove undecidability results to show the limits of these models. A prototype tool in OCaml implements the algorithms of this thesis. Despite the high complexity of the problems, this tool allows to prove liveness properties of communication protocols such as the Alternating Bit Protocol and Pachl's protocol.

Keywords : Formal verification, Lossy Channel Systems, Probabilities.

Remerciements

« Il faut toujours remercier l'arbre à karité sous lequel
on a ramassé de bons fruits pendant la bonne saison. »
Ahmadou Kourouma

Ils sont nombreux ceux qui m'ont aidée et soutenue pendant mes années de thèse. J'aimerais leur exprimer ma reconnaissance ici.

Sans eux je n'aurais pas soutenu ma thèse. Je remercie les membres de mon jury d'avoir accepté de faire partie de cette commission et d'avoir fait le déplacement pour ma soutenance. Merci en particulier à Jean-François Raskin et Igor Walukiewicz pour avoir relu mon manuscrit. Je suis très heureuse qu'ils aient accepté d'être mes rapporteurs. Merci également aux autres relecteurs minutieux, chercheurs de typos professionnels que sont Philippe et Manu. Je tiens à adresser un immense merci à Philippe qui m'a épaulée pendant ces trois années et demie (de DEA puis de thèse) qui furent mes premiers pas en recherche. Je le remercie en particulier pour sa disponibilité, ses conseils avisés mais aussi pour ses métaphores, toujours inattendues. Ich danke auch Christel, die meiner Doktorarbeit viele neue impulse gegeben hat und die so nett war, extra aus Dresden zu meiner Verteidigung zu kommen.[1]

Tous les membres du LSV que j'ai cotoyés au cours de mes quelques mois de DEA et de mes trois années de thèse sont aussi à remercier. Il règne dans ce laboratoire une excellente ambiance, et il était très agréable de la partager. Je profite donc de ce manuscrit pour remercier les LSVistes, et parmi eux : Tali, en mode Mama, qui a repris l'organisation des traditionnels goûters ; François, mon tuteur de monitorat, qui a eu la gentillesse de faire des délicieux baklavas pour mon pot de thèse ; tous ceux qui m'ont apporté une aide technique pendant mon séjour au LSV (en première ligne Pôti pour les questions générales et Nico pour les questions LaTeX) ; enfin ceux qui avec qui j'ai partagé mon bureau, par ordre d'apparition : Alexandre, Yu, Mathieu, Pascal, Ghassan et Thomas.

Y en a qui disent que LSV signifie aussi « Loisir, Sport et Vacances »... Pour ne pas les décevoir, je souhaite remercier tous les sportifs (et ils sont nombreux) que j'ai croisés pendant ma thèse. Tout d'abord les sportifs du LSV : que ce soit pour la piscine, le squash, le foot, ou même le tennis et le ping-pong, on trouvait toujours des motivés avec qui il était agréable de faire un peu de sport entre midi et deux. Merci donc en particulier à Pôti, Fabrice, Arnaud, Benedikt, Pascal, et Nico. Ensuite viennent les filles de l'équipe de hand de l'ENS avec qui j'ai passé des moments... très bleus ! Merci à toutes et principalement à Fanny, Cham et Plouf, fidèles parmi les fidèles. Je remercie aussi les adeptes d'un sport peu connu (le tennis-ballon) et surtout Greg, Fanny, Pierre, Alexia et Yoann. Je profite enfin de l'occasion pour remercier les membres de l'ensemble vocal que j'ai intégré il y a deux ans grâce à Claudine. Pour cela et pour nos discussions en rapport ou non avec la chorale, je la remercie.

Le travail, le sport, la chorale, ca fait déjà pas mal... et le reste ? Merci aux amis que je n'ai pas encore cités : Claire et Mathieu pour les pains de thon et autres tartes au citron, Jamie pour les recettes de glaces et les conseils tricotesques, Claire, Nicolas et Laure, et indépendamment Frédérique et Fabrice pour les repas-jeux, Manu pour les échanges de polars, Élise et son large sourire pour les moments Père Noyel, sans oublier Aline et ses fameuses alinades.

Enfin merci à ma famille qui m'entoure depuis toutes ces années. Merci en particulier à Cathy et Jacques pour leur pension complète accueillante aux Buttes Chaumont puis à

[1]Benedikt, danke für deine Hilfe als ich diesen Satz geschrieben habe :-)

Marseille, et pour leur aide précieuse pour mon pot de thèse. Merci à mes frères Nicolas et Benoît ainsi qu'à Vérane que j'apprécie de retrouver régulièrement, par exemple autour d'éclairs à la vanille. Enfin merci à mes parents pour la touche tourangelle de mon pot de thèse, mais surtout pour leur soutien et leurs encouragements permanents.

Table des matières

Introduction

La vérification : une nécessité Les systèmes informatisés sont de plus en plus présents dans notre société. On trouve par exemple des systèmes embarqués dans les moyens de transport ou de télécommunication. Ces applications mettent en œuvre des programmes ayant des propriétés définies dans un cahier des charges, qui ne coïncide par forcément avec l'implémentation. Dans certains domaines, les comportements inattendus de logiciels peuvent avoir des conséquences dramatiques, tant sur le plan économique que sur le plan humain. D'autre part, la complexité croissante des algorithmes utilisés fait qu'il est de plus en plus difficile de se convaincre de la fiabilité des programmes. Pour toutes ces raisons, il est de nos jours nécessaire de concevoir des méthodes pour prouver le bon fonctionnement des systèmes informatiques.

Les méthodes formelles Nous nous intéressons plus particulièrement aux méthodes dites formelles, qui cherchent à garantir la fiabilité des systèmes en passant par une modélisation mathématique de ceux-ci. Il y a trois grandes familles de méthodes formelles : le test, la démonstration automatique et le model-checking (ou vérification de modèles). Le test consiste à générer des scénarios d'exécution du système, et à vérifier que sur ces cas particuliers, le système suit le comportement attendu. La démonstration automatique a pour but de déduire mathématiquement les propriétés du système pour pouvoir s'assurer de son bon fonctionnement. Enfin le model-checking permet de vérifier automatiquement qu'un système satisfait une propriété donnée. Le point commun de ces méthodes est de fournir un cadre formel permettant d'augmenter la confiance dans la fiabilité des programmes. Par la suite, nous nous plaçons dans le cadre du model-checking, dont le principe est détaillé ci-dessous.

Le «model-checking» Le model-checking (ou vérification de modèles), est une méthode de validation des systèmes informatiques qui consiste à vérifier automatiquement si le modèle $\mathcal{S}$ d'un système satisfait une propriété ϕ, ceci étant noté $\mathcal{S} \models \phi$. Étant donné un programme informatique et une propriété comportementale pour ce programme, l'approche par model-checking passe tout d'abord par la modélisation à la fois du programme et de la propriété. Dans un deuxième temps, un algorithme est appliqué à ces deux objets pour déterminer si la modélisation du système satisfait la spécification de la propriété. La mise en place de techniques de model-checking comporte donc trois aspects : développer des modèles pour les systèmes informatiques (tenant compte des spécificités du programme à analyser), trouver des formalismes adaptés à l'expression des propriétés à vérifier, et enfin concevoir des algorithmes et des outils pour tester si le modèle satisfait la spécification. Dans cette thèse, nous définissons de nouveaux modèles et algorithmes pour la vérification de protocoles de communication. Un des intérêts du model-checking est d'utiliser, côté spécification un langage précis, par opposition aux propriétés exprimées en langage naturel que l'on trouve par exemple dans un cahier des charges. D'autre part, en comparaison avec le test le model-checking possède l'avantage de considérer toutes les exécutions possibles d'un système. Cette exhaustivité est particulièrement appréciable dans le cas où à la fois du non-déterminisme et des probabilités interviennent, car ces deux notions multiplient le nombre de tests qu'il faudrait générer. Enfin, le formalisme du model-checking offre un cadre unique pour chercher à prouver la correction ou la fiabilité d'un système ainsi que pour évaluer ses performances.

Les protocoles de communications Dans cette thèse, nous nous intéressons à un certain type de systèmes informatiques, les protocoles de communication asynchrones. Ces derniers peuvent être modélisés sous la forme de systèmes communicants : plusieurs entités s'envoient (et reçoivent) des messages et changent d'états en fonction de ces communications. Plus particulièrement, nous considérerons des systèmes communicants par canaux non fiables, pouvant ainsi exprimer des erreurs dans la transmission des messages (perte, corruption, insertion, etc ...). De façon surprenante, ces hypothèses de non-fiabilité permettent d'établir des résultats de décidabilité, et donc des algorithmes, qui n'existent pas dans le cas parfait (sans erreur de transmission). Par exemple, des techniques symboliques ont permis de vérifier automatiquement certaines propriétés de sûreté de protocoles asynchrones conçus pour résister aux pertes de message.

Les contributions de cette thèse Nous détaillons les contributions de cette thèse, partie par partie.

Première partie : Dans la première partie, nous faisons des rappels sur les automates communicants et les automates communicants à pertes (ou Lossy Channel Systems). Nous introduisons également une nouvelle représentation pour les ensembles de configurations de ces derniers. Ces ensembles sont appelés régions régulières puisqu'ils sont constitués de configurations dont les contenus des canaux forment des langages réguliers. Nous donnons également une extension des Lossy Channel Systems : les LCS étendus. Dans ce nouveau modèle, les règles de transition sont gardées, précisément par des régions régulières. Cela permet de rendre une transition tirable ou non, selon les messages qui se trouvent dans les canaux. Par exemple, on peut tester le vide d'un canal, ou faire des tests d'occurrence d'un message donné pour autoriser le franchissement d'une transition. Certains protocoles de communication utilisent de tels tests dans leur spécification ; les LCS étendus offrent alors la possibilité de mieux modéliser les protocoles, en supprimant les abstractions faites quand on modélise un choix exclusif par du non déterminisme.

La deuxième contribution du Chapitre 1 consiste à établir, dans un cadre général, une condition suffisante pour la convergence en temps fini de points fixes définis par des termes d'un μ-calcul (et exposée dans [BBS06b]). Plus précisément, nous nous plaçons dans la situation où un système infini est muni d'un bel ordre. Sur l'ensemble des configurations de ce système, nous définissons une algèbre des régions : c'est-à-dire un ensemble de régions (qui sont des ensembles de configurations particuliers) et des opérateurs monotones sur ces régions. Dans la suite, on s'intéresse en particulier aux algèbres effectives, *i.e.* pour lesquelles les opérateurs sont effectifs. Dans ce cadre très général d'algèbre des régions effective, nous établissons alors un critère de convergence des termes définis par points fixes (éventuellement imbriqués). Ce critère repose sur les propriétés du bel ordre entre les configurations, en particulier la convergence en temps fini des suites croissantes d'ensembles fermés vers le haut.

Les LCS étendus avec les régions régulières et, pour opérateurs, les opérateurs booléens, les prédécesseurs, et les opérateurs de clôture (vers le haut et vers le bas) forment une instance de l'algèbre des régions, sur lequel notre théorème général est valable. Comme application de ce résultat, on obtient donc des résultats (nouveaux ou déjà connus) de convergence de point fixes pour les LCS, et ce, de façon élégante. Par exemple, on redémontre la décidabilité du problème de l'accessibilité (et même la calculabilité de l'ensemble des prédécesseurs), de l'accessibilité contrainte, et on prouve la décidabilité d'une nouvelle notion : l'accessibilité sans risque. Ces résultats sur les ensembles calculables pour les LCS nous seront très utiles pour la vérification des PLCS et des NPLCS, deux modélisation des protocoles de communications asynchrones, qui ajoutent une notion probabiliste de correction.

Deuxième Partie : Nous définissons un nouveau modèle de PLCS (Probabilistic Lossy Channel Systems), qui diffère quelque peu des modèles existants. La modélisation des automates communicants à pertes par des chaînes des Markov constitue, en particulier vis à vis du traitement des pertes, une amélioration par rapport à la modélisation exclusivement non déterministe. En effet, les pertes de messages sont vues de façon naturelles comme des fautes qui se produisent de façon aléatoire. Le modèle non déterministe est trop pessimiste : il se peut que tous les messages soient perdus. Les probabilités permettent d'évacuer ces comportements marginaux puisqu'ils forment des ensembles négligeables.

Le modèle de PLCS que nous définissons [BS03, ABRS05] est un peu différent du modèle précédent, introduit pas Purushothaman Iyer et Narasimha [PN97]. Il est à nos yeux plus réaliste pour deux raisons. Tout d'abord, dans notre modèle, la probabilité qu'un message (non fixé) soit perdu est plus grande si les canaux contiennent beaucoup de messages, ce qui est réaliste en pratique. D'autre part les résultats que nous obtenons pour vérification sont indépendants du taux de perte, ce qui semble raisonnable quand celui-ci n'est pas connu, et seulement estimé. En particulier, on montre que quelque soit le taux de perte, l'ensemble des configurations où les canaux sont vides constitue un attracteur pour le système. Cela signifie que presque toutes les exécutions du système visitent infiniment souvent cet ensemble de configurations.

Pour prouver l'existence de l'attracteur fini, nous avons défini un critère, qui permet de rendre les preuves plus élégantes, et surtout moins *ad hoc*. Brièvement, si l'on peut partitionner l'ensemble des configurations d'une chaînes de Markov en tranches numérotées par les entiers naturels et telles que à chaque étape, l'espérance de la prochaine tranche est moindre, alors la tranche 0 est un attracteur. Nous avons qualifié de telles chaînes de Markov de «orientées à gauche» puisque l'on peut s'imaginer les tranches les unes à côté des autres, la tranche 0 étant le plus à gauche.

L'étude des chaînes de Markov avec attracteur fini possède des points communs avec l'étude des chaînes de Markov finies. En particulier, les valeurs des probabilités sur les transitions n'influent pas pour la vérification qualitative de propriétés : seule la topologie du graphe sous-jacent importe. Nous donnons des algorithmes pour la vérification de propriétés d'accessibilité et d'accessibilité répétée dans les chaînes de Markov avec attracteur fini, puis nous les appliquons aux PLCS. Ces résultats permettent de prouver la décidabilité de la vérification qualitative de formules du temps linéaire pour les PLCS.

Troisième Partie : Dans une troisième partie, nous introduisons un nouveau formalisme, celui des NPLCS (Nondeterministic Probabilistic Channel Systems). Les NPLCS fournissent un bon modèle pour les protocoles de communication, en modélisant les pertes de façon probabiliste et les choix entre les actions du système de façon déterministe. Au lieu des chaînes de Markov (comme pour les PLCS), on s'intéresse donc à des processus de décisions markoviens (ou encore chaînes de Markov concurrentes).

La vérification des processus de décision markoviens est plus complexe que celle des chaînes de Markov. Tout d'abord, on ne peut pas définir directement d'espace de probabilité sur les processus de décision markoviens. Il est nécessaire pour cela de fixer les choix non déterministes par le biais de ce que l'on appelle un adversaire. Une fois un adversaire fixé, un processus de décision markovien devient une chaîne de Markov (encore à espace d'états dénombrable). Les questions qui se posent alors sont du types : «quel que soit l'adversaire, la probabilité de vérifier une formule donnée est-elle 1 ?» Il existe en réalité quatre variantes de vérification qualitative selon que l'on veut vérifier une propriété presque sûrement, avec probabilité strictement positive, avec probabilité strictement inférieure à 1 ou avec probabilité nulle.

Le premier résultat que nous établissons est l'indécidabilité de la vérification de formules du temps linéaire pour les quatre variantes. En utilisant la négation, il suffit en fait de le prouver pour deux cas (par exemple «presque sûrement» et «avec probabilité positive»). Cependant, pour certaines classes de formules (par exemple les propriétés d'accessibilité), les quatre problèmes sont décidables. De plus, la décidabilité est encore vraie dans ces cas pour des adversaires restreints : les adversaires à mémoire finie. Comme les preuves d'indécidabilité utilisent fortement des adversaires à mémoire infinie, nous menons une nouvelles étude, cette fois-ci en se restreignant aux adversaires à mémoire finie. Il en ressort que les problèmes de vérification qualitative de formules du temps linéaire sont décidables pour les NPLCS en présence d'adversaires à mémoire finie. On le montre en plusieurs temps : tout d'abord on s'intéresse à des propriétés simples (accessibilité et accessibilité répétée), puis on montre la décidabilité pour des formules de Streett. Dans tous les cas, on écrit l'ensemble des configurations à partir desquelles tous les adversaires vérifient une propriété donnée pour un critère qualitatif, sous la forme d'un terme gardé du μ-calcul. La convergence découle alors du théorème que nous avons établi dans la première partie. Pour obtenir la décidabilité pour les formules du temps linéaire en général, on utilise une méthode classique : à partir d'une formule φ, on construit un automate fini avec conditions d'acceptation de Streett dont le langage accepté est constitué des exécutions satisfaisant φ ; le produit d'un NPLCS avec l'automate de Streett est un nouveau NPLCS qui a les bonnes propriétés suivantes : la mesure des exécutions du nouveau NPLCS qui vérifient une propriété de Streett est la même que la mesure des exécutions de l'ancien NPLCS qui vérifient φ. On réduit donc la vérification de formules du temps linéaire à la vérification de propriétés de Streett.

Nous avons étendu ces résultats à des adversaires (toujours à mémoire finie) équitables. Par équitables, on entend que l'ensemble des exécutions non équitables qui sont générées est négligeable. Les propriétés d'équité sont des propriétés d'équité forte de la forme : si infiniment souvent un ensemble de transitions est tirable, alors infiniment souvent une des transitions de cet ensemble sera tirée. En conclusion, pour les formules du temps linéaire et des adversaires équitables, la vérification qualitative est décidable pour les NPLCS.

<u>Quatrième Partie :</u> À partir des techniques exposées dans la troisième partie, nous avons développé un prototype de model-checker pour les NPLCS. Nous donnons des détails sur l'implémentation et décrivons les résultats obtenus dans la quatrième Partie. Ce prototype, implémenté en OCaml, nous a permis de vérifier des propriétés de sûreté, et de vivacité pour deux protocoles de communications : le protocole du Bit Alterné et le protocole de Pachl. Nous avons prouvé le bon fonctionnement de ces deux protocoles modélisés par des NPLCS, sous certaines hypothèses d'équité (codées dans la formule ou par l'intermédiaire des adversaires).

Première partie

Lossy Channel Systems

Dans cette partie, nous introduisons les automates communicants, ou *Channel Systems*, puis les automates communicants à pertes (*Lossy Channel Systems*, “LCS”) qui sont des automates communicants pour lesquels on considère que les canaux de communication ne sont pas fiables et peuvent perdre des messages. On fournit un cadre général pour l’étude des systèmes infinis qui est particulièrement adapté aux LCS. Les techniques de vérification présentées ici serviront dans les parties suivantes, c’est-à-dire lors de l’étude des LCS probabilistes, et des LCS probabilistes et non déterministes. Nous rappelons les définitions classiques des systèmes communicants et systèmes communicants à pertes et énonçons certains résultats qui nous seront utiles dans la suite. On profite de ces rappels pour définir une nouvelle sémantique opérationnelle qui généralise celles qui existaient précédemment. D’autre part, on donne un cadre général pour la convergence de points fixes qui apparaissent dans les problèmes de vérification. Ce résultat s’applique en particulier à la vérification des LCS, mais aussi, comme on le verra dans les parties suivantes, à la vérification des LCS avec pertes probabilistes.

Chapitre 1

Automates communicants

Sommaire

Les automates communicants (*channel systems*) sont des systèmes composés d'automates finis qui peuvent communiquer par l'intermédiaire de canaux non bornés. Les communications se font de façon asynchrone et l'ordre des messages est toujours préservé, *i.e.* les canaux sont FIFO (*first in first out*). Ainsi, les messages sont reçus dans l'ordre où ils sont envoyés. Ce modèle permet de décrire naturellement des protocoles de communications entre plusieurs entités. Il est d'ailleurs à la base de la sémantique de certains langages de spécification de protocoles tels que SDL [ITU99] et Estelle [BD87]. Le système représenté Figure 1.1 constitue un exemple d'automate communicant. Deux composants, le "serveur" et le "client", (représentés

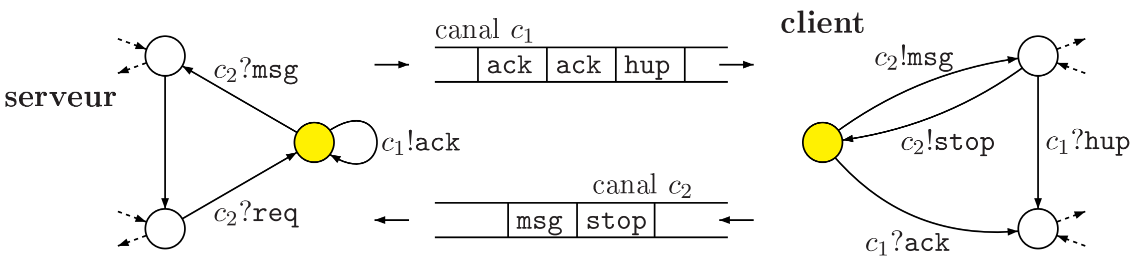

FIG. 1.1 – Un exemple d'automates communicants

de part et d'autre des canaux) communiquent par l'échange de messages au travers des canaux c_1 et c_2. Le serveur et le client peuvent changer d'état de contrôle en envoyant ou en recevant des messages. L'action consistant à écrire un message `msg` en queue du canal c_2 est notée c_2!`msg` alors que celle correspondant à la lecture du message `ack` en tête du canal c_1 est notée c_1?`ack`.

1.1 Le modèle des automates communicants

Toutes les définitions seront illustrées sur l'exemple qui suit. Il s'agit d'un automate communicant composé d'un émetteur et d'un récepteur qui implémentent le protocole du Bit Alterné [BSW69]. Ce protocole a pour but de pallier les pertes de messages en envoyant à plusieurs reprises les données et les accusés de réception, et utilise un bit (de valeur 0 ou 1) pour distinguer les messages.

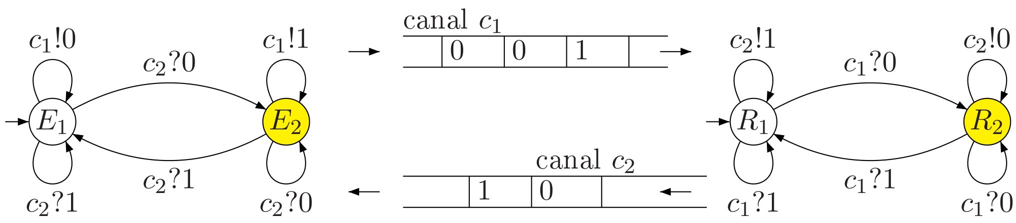

FIG. 1.2 – Une modélisation du protocole du Bit Alterné

La Figure 1.2 représente les deux composants (émetteur et récepteur) et les canaux c_1 et c_2 contenant des messages ; c_1 est utilisé pour les données et c_2 pour les accusés de réception.

1.1.1 Canaux parfaits

Dans la définition formelle d'un automate communicant, on considère un unique automate fini dont le comportement est déterminé par les messages qu'il peut écrire et lire dans des canaux, qu'il utilise comme des files FIFO. Cette hypothèse n'est en aucun cas une restriction, puisqu'il suffit à partir de plusieurs automates de construire le produit pour obtenir un seul processus.

Dans la Figure 1.3, on a réalisé le produit des deux composants, pour obtenir un unique automate fini. De plus, pour simplifier le schéma, on a représenté plusieurs boucles en une seule en écrivant de multiples opérations sur une unique boucle.

Définition 1.1 (Automate communicant - CS)**.** *Un* automate communicant *(ou* Channel System*, CS en abrégé) est un n-uplet* $\mathcal{L} = \langle S, \mathsf{C}, \mathsf{M}, \Delta\rangle$ *composé des éléments suivants :*

- $S = \{s, \dots\}$ *un ensemble fini d'états de contrôle,*
- $\mathsf{C} = \{c_1, \dots\}$ *un ensemble fini de canaux,*
- $\mathsf{M} = \{m, \dots\}$ *un alphabet fini de messages, et*
- $\Delta = \{\delta, \dots\}$ *un ensemble fini de règles de transition.*

Les règles de transition forment un sous-ensemble de $S \times (\{\mathsf{C} \times \{?, !\} \times \mathsf{M}\} \cup \{\surd\}) \times S$. *Si* $\delta = (s, op, s') \in \Delta$ *est une règle, op est appelé l'*opération *de* δ *et on note* $\delta : s \xrightarrow{op} s'$. *On écrira plus simplement* $c!m$ *(resp.* $c?m$*) pour* $(c, !, m)$ *(resp.* $(c, ?, m)$*). On détaille ci-dessous la signification des différentes opérations.*

- $c!m$ *envoyer le message m sur le canal c,*
- $c?m$ *lire le message m sur le canal c,*
- $\surd$ *action interne d'un des processus.*

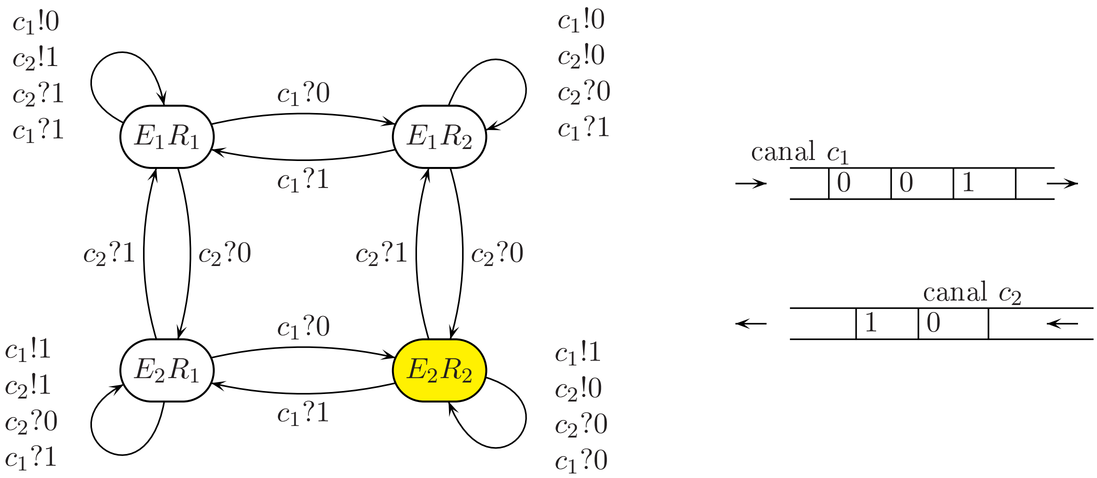

FIG. 1.3 – Le Bit Alterné sous forme d'automate communicant

Sémantique opérationnelle Dans la suite on considère un automate communicant $\mathcal{L} = \langle S, \mathsf{C}, \mathsf{M}, \Delta \rangle$ fixé.

On appelle *configuration* de $\mathcal{L}$ un couple $\sigma = (q, \mathbf{w})$ constitué d'un état de contrôle $s \in S$ et d'une valuation des canaux : $\mathbf{w} : \mathsf{C} \to \mathsf{M}^*$. Une configuration représente l'état du système, à la fois l'état de contrôle dans lequel se trouve l'automate fini, et les contenus des canaux. Dans la suite, on notera pour simplifier ε à la fois pour le mot vide et pour désigner la valuation ayant tous les canaux vides. L'ensemble de toutes les configurations est noté Conf. Dans l'exemple de la Figure 1.3, la configuration dans laquelle se trouve l'automate communicant est $(E_2R_2, (100, 10))$ c'est-à-dire que l'état de contrôle est E_2R_2 (grisé) le contenu du canal c_1, 100 et le contenu du canal c_2 est 10.

Définition 1.2 (Règle tirable)**.** *Une règle de transition δ : (s, op, s') est* tirable *dans la configuration $\sigma = (t, \mathbf{w})$ si*

- *premièrement $s = t$ et*
- *l'opération op est réalisable à partir de $\mathbf{w}$; formellement*

  écriture ou action interne *aucune contrainte sur $\mathbf{w}$*

  lecture *si $op = c?m$, alors il existe $w \in \mathsf{M}^*$ tel que $\mathbf{w}(c) = mw$, c'est-à-dire qu'un message m se trouve en tête du canal c.*

L'ensemble des règles de transition de Δ tirables dans une configuration σ donnée est noté $\Delta(\sigma)$. Toujours sur l'exemple du modèle du bit alterné, et pour la configuration courante, $\Delta(E_2R_2, (100, 10)) = \{E_2R_2 \xrightarrow{c_1?1} E_2R_1, E_2R_2 \xrightarrow{c_1!1} E_2R_2, E_2R_2 \xrightarrow{c_2!0} E_2R_1, E_2R_2 \xrightarrow{c_2?1} E_1R_2, \}$.

Remarque 1.3. *On fera souvent l'hypothèse qu'il n'y a aucune configuration bloquante, et donc que $\Delta(\sigma) \neq \emptyset$ pour toute configuration $\sigma \in \mathsf{Conf}$. Cela revient à supposer en particulier que pour chaque état de contrôle $s \in S$, il existe une règle de transition $\delta = (s, op, s')$ où op est une opération du type écriture ou action interne puisque ce sont les seules qui sont tirables lorsque les canaux sont vides.*

À un automate communicant, on associe un système de transitions, dont les états sont les configurations de Conf, et les transitions sont décrites ci-dessous. On note $\to_{\text{perf}}$ la relation de transition[1]. Soient σ et τ deux configurations. On note $\delta : \ \sigma \rightarrow_{\mathrm{perf}} \tau$ (ou encore $\sigma \xrightarrow{\delta}_{\mathrm{perf}} \tau$) si δ est tirable dans σ, et τ est le résultat de l'application de la règle de transition δ à la configuration σ. Détaillons ce dernier point. Supposons que $\delta = (s, op, t)$, $\sigma = (s, \mathbf{w})$ et $\tau = (t, \mathbf{v})$ (l'état de contrôle de σ est nécessairement s pour que δ soit tirable). Les contenus des canaux v après transition sont obtenus à partir de $\mathbf{w}$ et en fonction de l'action op comme on s'y attend :

<u>écriture $c!m$:</u> $\mathbf{v}$ est obtenu à partir de $\mathbf{w}$ en écrivant m à la fin du contenu du canal c :

$$\mathbf{v}(d) \stackrel{\mathrm{def}}{=} \begin{cases} \mathbf{w}(d) & \text{si } d \neq c \\ \mathbf{w}(d)m & \text{sinon.} \end{cases}$$

<u>lecture $c?m$:</u> $\mathbf{v}$ est obtenu à partir de $\mathbf{w}$ en lisant m en tête du canal c :

$$\mathbf{v}(d) \stackrel{\mathrm{def}}{=} \begin{cases} \mathbf{w}(d) & \text{si } d \neq c \\ w & \text{sinon, et si } \mathbf{w}(d) = mw. \end{cases}$$

<u>action interne $\surd$:</u> le contenu des canaux est inchangé, $\mathbf{v} = \mathbf{w}$.

On note $\mathsf{ST}_{\mathcal{L}} = (\mathsf{Conf}, \rightarrow_{\mathrm{perf}})$ le système de transitions ainsi obtenu. Telle que nous l'avons définie, la relation de transition $\rightarrow_{\mathrm{perf}}$ est un sous-ensemble de $\mathsf{Conf} \times \Delta \times \mathsf{Conf}$. Parfois, on la verra comme un sous-ensemble de $\mathsf{Conf} \times Op \times \mathsf{Conf}$ où Op est l'ensemble des opérations.

Prédécesseurs et successeurs On définit les prédécesseurs et successeurs d'une configuration et d'un ensemble de configurations comme habituellement. Tout d'abord, les prédécesseurs et successeurs d'une configuration σ par une règle de transition δ sont donnés par :

$$Pre[\delta](\sigma) \stackrel{\mathrm{def}}{=} \{\tau \in \mathsf{Conf} | \tau \xrightarrow{\delta}_{\mathrm{perf}} \sigma\}$$
$$Post[\delta](\sigma) \stackrel{\mathrm{def}}{=} \{\tau \in \mathsf{Conf} | \sigma \xrightarrow{\delta}_{\mathrm{perf}} \tau\}$$

Ensuite, pour $A \subseteq \mathsf{Conf}$,

$$Pre[\delta](A) \stackrel{\mathrm{def}}{=} \bigcup_{\sigma \in A} Pre[\delta](A) \quad \text{et} \quad Post[\delta](A) \stackrel{\mathrm{def}}{=} \bigcup_{\sigma \in A} Post[\delta](A),$$
$$Pre(A) \stackrel{\mathrm{def}}{=} \bigcup_{\delta \in \Delta} Pre[\delta](A) \quad \text{et} \quad Post(A) \stackrel{\mathrm{def}}{=} \bigcup_{\delta \in \Delta} Post[\delta](A).$$

Pouvoir d'expression Le modèle des automates communicants possède la puissance des machines de Turing [BZ83]. Plus précisément, les automates communicants permettent de simuler des machines de Turing en temps quadratique. Un canal de communication est suffisant pour simuler un ruban de machine de Turing ; les écritures et lectures en milieu de ruban nécessitent de permuter circulairement les messages puisqu'on ne peut qu'écrire en queue et lire en tête. Ceci explique le coût quadratique du codage.

Le fait qu'on puisse simuler une machine de Turing par un automate communicant entraîne l'indécidabilité de tous les problèmes de vérification non triviaux, conformément au Théorème de Rice. Par exemple, l'accessibilité (savoir si une configuration donnée - états des processus

[1]L'indice perf met l'accent sur le fait que les transitions sont parfaites, c'est-à-dire sans pertes, par opposition avec le cas des *Lossy Channel Systems*.

et contenu des canaux - est accessible à partir d'une configuration initiale) est un problème indécidable.

Des restrictions sur le modèle permettent de simplifier les problèmes et en particulier entraînent des résultats de décidabilité. C'est bien sûr le cas si l'on borne la taille maximale des canaux. On obtient alors un système ayant un nombre fini de configurations et l'accessibilité devient décidable. Le problème est alors PSPACE-complet. Néanmoins, il est commode de conserver des canaux possiblement infinis pour, par exemple, prendre en compte les délais arbitraires de transmission des messages. Dans le cadre de canaux à capacité infinie, une façon de restaurer la décidabilité de certaines propriétés est de se restreindre aux systèmes communicants pour lesquels l'ensemble d'accessibilité est régulier [Pac87], c'est-à-dire dans le cas où, pour tout état de contrôle l'ensemble des contenus de canaux possibles à partir d'une configuration initiale fixée constitue un langage régulier. Il se trouve que cette propriété de régularité est fréquente en pratique. C'est le cas par exemple pour le modèle du bit alterné de la Figure 1.3 page 21, quelque soit la configuration initiale. Cependant l'algorithme de décision associé n'est pas utilisable en pratique.

Une façon de "simplifier" le modèle est de considérer que les canaux ne sont pas totalement fiables, et peuvent perdre des messages. Cette hypothèse est de plus pertinente dans le cas de modélisation de protocoles de communication, car les pertes de messages sont tout à fait réalistes dans ce domaine. Dans le modèle des automates communicants par canaux à pertes, certains problèmes, dont l'accessibilité, sont décidables alors qu'ils sont indécidables dans le cas de canaux parfaits. Nous le verrons en détail dans la suite.

1.1.2 Canaux à pertes

On s'intéresse à présent à un modèle d'automates communicants avec canaux non fiables. Nous désignerons dans cette thèse un tel système par le terme de LCS, pour *Lossy Channel System*. Le modèle des LCS permet de décrire de façon réaliste des protocoles qui justement ont pour but d'établir une communication en présence de perturbations, que ce soient des pertes, des altérations ou encore des insertions de messages. Le principal type d'erreurs que nous considérons est la perte de messages. Ponctuellement (dans la sous-section 3.5.1), on considérera également d'autres comportements défectueux, comme la duplication, la corruption, ou l'insertion de messages.

Afin de formaliser les pertes de messages, on commence par introduire une relation, appelée relation de sous-mot.

Relation de sous-mot

Les pertes de messages sont formalisées par l'intermédiaire de l'ordre sous-mot sur les contenus des canaux. Les propriétés de cette relation sont à l'origine des résultats positifs pour les systèmes à canaux non fiables.

Définition 1.4 (Sous-mot)**.** *Soit Σ un alphabet fini. Pour $x, y \in \Sigma^*$, on dit que x est un* sous-mot *de y, noté $x \sqsubseteq y$ s'il existe $y_0, \cdots, y_n \in \mathsf{M}^*$ tels que*

$$x = a_1 \cdots a_n \text{ et } y = y_0 a_1 y_1 \cdots a_n y_n \text{ avec } a_i \in \Sigma.$$

Autrement dit, x est sous-mot de y s'il peut être obtenu à partir de y en «effaçant» des lettres, non forcément contiguës. La relation sous-mot est réflexive, transitive et antisymétrique, donc $\sqsubseteq$ est une relation d'ordre. De plus,

Théorème 1.5 (Higman [Hig52])**.** *La relation $\sqsubseteq$ est un bel-ordre partiel.*

Cela signifie que de toute suite infinie $(x_i)_{i\in\mathbb{N}}$ de mots sur l'alphabet Σ, on peut extraire une sous-suite infinie croissante. Ou encore que tout ensemble de mots possède un nombre fini d'éléments minimaux pour l'ordre $\sqsubseteq$. On renvoie le lecteur à [Kru72] pour ces deux caractérisations équivalentes.

Nous introduisons à présent les notions de fermeture vers le haut (ou vers le bas), et de noyau fermé vers le haut (ou vers le bas) pour un ensemble de mots. Ces définitions sont générales pour les ensembles munis d'une relation d'ordre. Soit donc $(W, \sqsubseteq)$ un ensemble muni d'un bel ordre.

Définition 1.6 (Ensembles fermés particuliers)**.** *Soit $V \subseteq W$. On définit les ensembles suivants :*

fermeture vers le haut $C_\uparrow V \stackrel{\text{def}}{=} \{w \in W | \exists v \in V \ t.q.\ v \sqsubseteq w\}$

fermeture vers le bas $C_\downarrow V \stackrel{\text{def}}{=} \{w \in W | \exists v \in V \ t.q.\ w \sqsubseteq v\}$

noyau fermé vers le haut $K_\uparrow V \stackrel{\text{def}}{=} \{w \in V | \forall v \sqsupseteq w,\ v \in V\}$

noyau fermé vers le bas $K_\downarrow V \stackrel{\text{def}}{=} \{w \in V | \forall v \sqsubseteq w,\ v \in V\}$

On dit que $V \subseteq W$ est fermé vers le haut si $V = C_\uparrow V$. Symétriquement, V est fermé vers le bas si $V = C_\downarrow V$.

La fermeture vers le haut de V est le plus petit fermé vers le haut contenant V. Le noyau fermé vers le haut de V est le plus grand fermé vers le haut inclus dans V. La fermeture vers le bas et le noyau fermé vers le bas possèdent des caractéristiques symétriques :

Proposition 1.7. *Soit V un sous-ensemble quelconque de W. Alors,*
- $W \setminus K_\uparrow(V) = C_\downarrow(W \setminus V)$
- $W \setminus K_\downarrow(V) = C_\uparrow(W \setminus V)$

Notation 1.8. *On écrira fréquemment $\uparrow V$ au lieu de $C_\uparrow V$ pour désigner la fermeture vers le haut de V. Si $v \in W$, $\uparrow v$ dénote l'ensemble des éléments $w \in W$ tels que $v \sqsubseteq w$.*

Si $\sigma \in \mathsf{Conf}$, $\uparrow\sigma$ dénote l'ensemble des configurations τ telles que $\sigma \sqsubseteq \tau$.

Une conséquence du résultat de Higman est que les suites croissantes d'ensembles fermés vers le haut convergent en temps fini [Kru72]. Précisément,

Proposition 1.9. *Soit $L_0 \subseteq L_1 \subseteq L_2 \cdots$ une suite croissante d'ensembles fermés vers le haut. Alors il existe un indice $k \in \mathbb{N}$ tel que $L_k = \bigcup_{i\in\mathbb{N}} L_i$.*

On étend la relation de sous-mot (et les opérateurs $C_\uparrow$, $C_\downarrow$, $K_\uparrow$ et $K_\downarrow$) aux configurations en demandant pour avoir $\sigma \sqsubseteq \sigma'$, que les états de contrôle soient identiques, et que pour chaque canal le contenu dans σ est un sous-mot du contenu dans σ'. Formellement :

$$(s, \mathbf{w}) \sqsubseteq (t, \mathbf{v}) \text{ ssi } \begin{cases} s = t \ \text{ et,} \\ \mathbf{w}(c) \sqsubseteq \mathbf{v}(c) \quad \forall c \in \mathsf{C}. \end{cases}$$

Remarque 1.10. *La relation $\sqsubseteq$ sur les configurations est encore un bel ordre.*

Sémantique opérationnelle

La sémantique que nous choisissons pour les LCS suppose que les pertes de messages ont lieu à chaque étape après une action (écriture, lecture, opération interne) du système.

Nous faisons ce choix dans la sémantique, et nous le suivrons dans la suite, puisqu'il facilite la définition des modèles probabilistes.

La sémantique des automates communicants par des canaux à pertes est donnée de façon formelle ci-dessous. Soient σ et σ' deux configurations d'un automate communicant $\mathcal{L}$. On note $\sigma \rightarrow_{\text{loss}} \sigma'$ si $\sigma' \sqsubseteq \sigma$, c'est-à-dire si la configuration σ' peut être obtenue à partir de σ en perdant des messages. Les transitions dans $\mathcal{L}$ considéré comme un automate à pertes sont toutes celles de la forme $\sigma \rightarrow \tau'$ pour lesquelles il existe τ telle que $\sigma \rightarrow_{\text{perf}} \tau$ et $\tau \rightarrow_{\text{loss}} \tau'$. En particulier, une étape parfaite (sans perte) est possible, et donc le comportement d'un LCS englobe celui du système communicant parfait qui lui est associé.

On appelle *chemin* de $\mathcal{L}$ toute suite finie de configurations $\sigma_0, \cdots, \sigma_n$ telle que $\sigma_i \rightarrow \sigma_{i+1}$ pour $0 \leq i \leq n-1$. Enfin, on appelle *exécution* une suite infinie de configurations $\sigma_0, \sigma_1, \cdots$ telle que $\sigma_i \rightarrow \sigma_{i+1}$ pour $i \geq 0$.

La sémantique que nous choisissons d'adopter ici possède la propriété suivante concernant les prédécesseurs en une étape.

Proposition 1.11. *Soit A un ensemble quelconque de configurations.*

$$Pre(A) = Pre_{\text{perf}}(\uparrow A)$$

où Pre_{perf} dénote l'ensemble des prédécesseurs par une transition parfaite (sans perte).

Preuve : Par définition $Pre[\rightarrow] = Pre[\rightarrow_{\text{perf}} \rightarrow_{\text{loss}}] = Pre[\rightarrow_{\text{perf}}]Pre[\rightarrow_{\text{loss}}]$, c'est-à-dire que Pre est la composition de Pre_{perf} et Pre_{loss}. Comme Pre_{loss} coïncide avec l'opérateur $\uparrow$ de fermeture vers le haut, on obtient le résultat attendu. □

1.1.3 Variantes de LCS

On peut définir de plusieurs façons des automates communicants à pertes. Ces modèles diffèrent par deux aspects : la localisation des pertes (à quel endroit du canal peuvent elles se produire ?) et la granularité à la fois des actions et des pertes.

Dans un premier temps, un canal non fiable était modélisé par l'intermédiaire d'un automate additionnel qui modifiait les contenus des canaux en y lisant des messages. C'est par exemple l'approche de Zafiropulo *et al.* dans [ZWR+80].

Cette modélisation amène Finkel à définir les *completely specified protocols* [Fin94], pour lesquels les pertes de messages sont modélisées par l'ajout sur chaque état de contrôle de boucles de lecture $(s, c?m, s)$ pour chaque couple (c, m) composé d'un canal et d'un message. Ces ajouts reviennent à considérer que les messages peuvent être perdus quand ils sont en tête des canaux. Le modèle de Finkel, comme celui de Zafiropulo *et al.*, ont pour avantage de ne pas nécessiter l'introduction d'une nouvelle sémantique opérationnelle : les boucles de lectures permettent de représenter les pertes au sein du système parfait.

Abdulla et Jonsson [AJ96b, AJ93] innovent alors en choisissant de modifier la sémantique opérationnelle plutôt que l'ensemble des règles de transition. Ceci permet selon la sémantique choisie à la fois de simuler le modèle de Finkel, mais aussi de définir un modèle où les pertes de messages peuvent se produire partout dans les canaux (et plus seulement en tête). Cette nouvelle modélisation est une véritable avancée conceptuelle et est mathématiquement plus élégante.

Selon les résultats à prouver, certains formalismes sont plus faciles à manier que d'autres, même si au fond le comportement des systèmes est similaire. Par exemple lorsque Abdulla *et al.* prouvent comment simuler des canaux parfaits avec des canaux à pertes probabilistes [ABPJ05], ils utilisent un modèle pour lequel les pertes de messages ne se font qu'au moment où les messages sont envoyés dans les canaux (pertes à l'écriture). Une autre preuve d'indécidabilité, par Schnoebelen [Sch01], est faite pour le modèle *front lossy* où les pertes ne se produisent qu'en tête des canaux.

La granularité entre les actions et les pertes est aussi différente selon les modèles. Par granularité, on entend la composition d'une étape du système de transitions sous-jacent : nombre de pertes possible et position par rapport aux actions du système. Dans tous les cas, la relation de transition donnée par la sémantique opérationnelle est une combinaison des deux relations $\rightarrow_{\text{perf}}$ et $\rightarrow_{-1}$, où $\rightarrow_{\text{perf}}$ est la relation de transition parfaite (sans perte) et $\rightarrow_{-1}$ représente la perte d'un message. Parfois, les transitions atomiques sont soit une action, soit une perte d'un message : c'est le cas des travaux de Baier et Engelen [BE99] dans leur modèle probabiliste, ou de Finkel pour les *completely specified protocols*. Le plus souvent néanmoins les pertes sont regroupées en une relation $\rightarrow_{\text{loss}}=\rightarrow_{-1}^*$, exprimée par la relation sous-mot. Il reste alors à choisir leur position par rapport aux actions : avant et après les actions, seulement après comme dans cette thèse, etc...

D'autre modèles pourraient être étudiés, par exemple en supposant que si la communication est défectueuse elle l'est pour tous les messages présents dans le canal. On pourrait par exemple demander que, soit aucun, soit tous les messages sont perdus, ou encore, que tous les messages qui suivent un message perdu sont perdus dans la même étape. Pour ces variantes les résultats seraient sans doute similaires. Nous fixons un modèle général dans cette thèse et la plupart des résultats se transposeraient aux différents modèles évoqués ci-dessus.

Dans cette thèse, on opte donc pour une sémantique opérationnelle où un nombre arbitraire de pertes suivent les actions du système.

1.2 Automates communicants étendus

Les règles de transition des systèmes communicants étudiés jusqu'alors (à la fois dans la littérature et dans cette thèse) sont uniquement constituées d'actions sans pré-condition, ou garde, explicite. Bien sûr, les lectures imposent une pré-condition implicite, en effet il faut que le message m soit en tête du canal c pour pouvoir tirer une règle de type $s \xrightarrow{c?m} t$. Le fait de ne pas ajouter de garde sur les transitions vise à simplifier le modèle. Néanmoins, les protocoles de communication utilisent fréquemment des gardes qui par exemple testent les contenus des canaux avant de choisir une transition plutôt qu'une autre. La pré-condition la plus simple est un test du vide, du type $s \xrightarrow{c=\varepsilon?} t$ qui permet d'autoriser une transition seulement si le canal c est vide.

Pour être à la fois plus général, plus simple et plus expressif, on considère dans cette thèse des gardes régulières sur les contenus des canaux. Aussi, nous introduisons dans [BBS06b] un nouveau modèle, appelé GLCS (pour Generalized Lossy Channel Systems) ou *LCS étendu* dans cette thèse. Ce modèle permet d'exprimer à la fois les tests de vide et les tests d'occurrence, ces derniers étant utilisés par Ouaknine et Worrell [OW06] dans le cadre d'automates communicants avec erreurs d'insertion.

Définition 1.12 (LCS étendu)**.** *Un* LCS étendu *est un n-uplet $\mathcal{L} = (S, \mathsf{C}, \mathsf{M}, \Delta)$ où $S, \mathsf{C}, \mathsf{M}$ sont, comme pour les LCS, des ensembles finis d'états de contrôle, de canaux et de messages, et où l'ensemble Δ est constitué de règles de transition δ de la forme*

$$s \xrightarrow{G:op} s'$$

où $s, s' \in S$, op est une opération comme dans les LCS standards et G est un ensemble régulier de contenus de canaux, appelé garde.

La garde G est un sous-ensemble de $(\mathsf{M}^*)^{\mathsf{C}}$ donné par exemple sous la forme d'une expression régulière ou d'une contrainte logique. Soit $\sigma = (s, \mathbf{w})$ une configuration de $\mathcal{L}$, on écrira $\mathbf{w} \in G$ pour dénoter que $\mathbf{w}$ satisfait la garde G.

Exemple 1.13. *Comme on l'a déjà évoqué, les gardes permettent de tester la présence de messages dans les canaux, et donc de faire des tests de vide ou des tests d'occurrences d'un message donné. On peut également utiliser des gardes pour implémenter des choix exclusifs, ou des priorités entre les règles : si les contenus de canaux vérifient telle propriété, les règles possibles sont celles-là, sinon ce sont celles-ci, etc... Ceci permet de supposer sans pertes de généralité que les LCS étendus que nous considérons sont sans configuration bloquante. Il suffit pour cela d'ajouter un état de contrôle qui sera l'état puits, et d'ajouter des règles de transition vers cet état à partir des configurations bloquantes. Ces dernières peuvent être exprimées par des gardes régulières comme complément des configurations non bloquantes. Enfin, on peut à l'aide des gardes établir un ordre de priorité entre les gardes, toujours en utilisant des gardes exclusives entre les différentes règles de transition possibles.*

La sémantique opérationnelle d'un LCS étendu est proche de celle d'un LCS ordinaire; néanmoins pour qu'une transition soit tirable, il faut désormais que la garde soit vérifiée, c'est-à-dire que la configuration appartienne à la région G. Plus précisément, soit $\delta : s \xrightarrow{G:op} s'$ une règle de transition du LCS étendu $\mathcal{L}$. Alors,

$$\sigma = (s, \mathbf{w}) \xrightarrow{G:op} \tau \text{ ssi } \begin{cases} \mathbf{w} \in G & \text{et,} \\ \sigma \to \tau. \end{cases}$$

Remarque 1.14. *On notera simplement $s \xrightarrow{op} s'$ plutôt que $s \xrightarrow{G:op} s'$ dans le cas où G est la contrainte triviale : $G = (\mathsf{M}^*)^{\mathsf{C}}$.*

La sémantique des GLCS permet d'exprimer simplement les prédécesseurs et les successeurs à partir des mêmes opérateurs dans les LCS classiques (*i.e.*, sans garde) :

Proposition 1.15. *Soit $A \subseteq \mathsf{Conf}$ un sous-ensemble des configurations de $\mathcal{L}$.*

$$Pre[s \xrightarrow{G:op} s'](A) = G \cap Pre[s \xrightarrow{op} s'](A),$$
$$Post[s \xrightarrow{G:op} s'](A) = Post[s \xrightarrow{op} s'](G \cap A).$$

1.3 Regular model-checking pour les LCS

Dans le but d'étudier le comportement des systèmes, en faisant par exemple du *regular model-checking*, plusieurs représentations ont été proposées pour les ensembles de configurations des automates communicants. Le *regular model-checking* [BJNT00, KMM+01], est un cadre de vérification symbolique pour les systèmes à espace d'états infini. Il a été appliqué à de nombreuses classes de systèmes, allant des algorithmes distribués aux systèmes hybrides ou aux programmes manipulant des structures de données dynamiques. Le principe du *regular model-checking* est de considérer des ensembles «réguliers» de configurations et de les représenter de façon finie, par exemple par des automates finis, ou des expressions régulières. Les systèmes qui se prêtent au *regular model-checking* sont tels que si A est un ensemble régulier de configurations, alors $Post(A)$ (ou $Pre(A)$) est encore un ensemble régulier, que l'on peut calculer à partir de A. Puisque les ensembles réguliers sont fermés pour les opérations booléennes, on peut alors essayer de calculer l'ensemble des configurations accessibles depuis `Init` (*resp.* co-accessibles) par la procédure suivante qui est l'algorithme naturel utilisé dans le cas des systèmes à espace d'états fini.

```
A := Init
B := Post(A)
tant que A ≠ B
      faire A := B
            B := Post(B)
fin de tant que
retourner A
```

L'égalité de deux ensembles réguliers étant décidable, le calcul ci-dessus peut, en utilisant un test d'égalité permettre de savoir si la limite est atteinte en temps fini.

Dans le cas des systèmes infinis, la convergence n'est pas assurée. Il est rare que l'algorithme d'itération donné pour le calcul de Pre^* converge en temps fini [BFLS05]. C'est pourquoi actuellement des méthodes sont proposées pour calculer directement, ou deviner et vérifier, ou approcher (par une sur-approximation ou sous-approximation) l'ensemble $Post^*(A)$ [BLW03, Boi03, BHV04, HV05, BFLS05].

La première contribution de cette thèse est de donner un résultat général de convergence (en temps fini) pour les calculs de points fixes. Nous établissons ce résultat pour un ensemble de configurations quelconques (pas forcément celui d'un LCS). On commence par définir la notion d'algèbre des régions monotone et par isoler un fragment d'un mu-calcul pour lequel la convergence est assurée. Enfin, nous montrons comment appliquer ce théorème général au cas des LCS.

1.3.1 Algèbre des régions

Soit $(W, \sqsubseteq)$ un ensemble muni d'un bel ordre.

Une *région* est un sous-ensemble de W, c'est-à-dire un ensemble d'éléments de W.

Définition 1.16. *Un opérateur d'arité n, $o : 2^{W^n} \to 2^W$ est* monotone *si pour tous n-uplets $(V_1, \cdots, V_n)$ et $(V'_1, \cdots, V'_n)$,*

$$V_1 \subseteq V'_1 \text{ et } \cdots \text{ et } V_n \subseteq V'_n \Rightarrow o(V_1, \cdots, V_n) \subseteq o(V'_1, \cdots, V'_n)$$

L'opérateur o est ω-sup-continu *si pour tout suite croissante $(V_n)_{n\in\mathbb{N}}$ de sous-ensembles de W,*

$$\bigcup_{n\in\mathbb{N}} o(V_n) = o(\bigcup_{n\in\mathbb{N}} V_n)$$

Enfin, o est ω-inf-continu *si pour toute suite décroissante $(V_n)_{n\in\mathbb{N}}$ de sous-ensembles de W,*

$$\bigcap_{n\in\mathbb{N}} o(V_n) = o(\bigcap_{n\in\mathbb{N}} V_n)$$

Un résultat important (cf. par exemple [AN01]) établit que pour les fonctions ω-sup-continues (resp. ω-inf-continues), le plus petit point fixe (resp. plus grand point fixe) est atteint à ω.

Proposition 1.17. *Si ϕ est ω-sup-continue, alors* $[\![\mu X.\phi]\!]_{env} = \bigcup_{n\in\mathbb{N}}[\![\phi^n(\emptyset)]\!]$.
Si ϕ est ω-inf-continue, alors $[\![\nu X.\phi]\!]_{env} = \bigcup_{n\in\mathbb{N}}[\![\phi^n(W)]\!]$.

Pour continuer la définition de l'algèbre des régions qui permettra de manipuler des ensembles de configurations par le biais d'opérateurs monotones, on s'intéresse à présent à ces derniers. Les opérateurs sont ceux que l'on appliquera pour définir des procédures de vérification symbolique sur les ensembles de configurations.

Définition 1.18 (Algèbre des régions monotone)**.** *Soit $O = \{o_1, o_2, \cdots\}$ une famille d'opérateurs* monotones, *contenant les opérateurs d'arité un $C_\uparrow, C_\downarrow, K_\uparrow, K_\downarrow$ et les opérateurs d'arité zéro $\emptyset$ et W.*

On appelle algèbre des régions *engendrée par O, notée $\mathcal{R}_O$, le plus petit sous-ensemble de 2^W clos par les opérateurs de O. Chaque élément de $\mathcal{R}_O$ est appelé* région.

De façon équivalente, $\mathcal{R}_O$ est construit inductivement à partir des opérateurs d'arité nulle et en appliquant les opérateurs de O aux régions construites jusqu'alors.

Remarque 1.19. *Les opérateurs $C_\uparrow, C_\downarrow, K_\uparrow, K_\downarrow$ sont bien monotones.*

L'algèbre des régions engendrée par O est dite *effective* s'il existe des algorithmes implémentant les opérateurs de O et le test d'appartenance à une région (étant donnés $w \in W$ et $R \in \mathcal{R}_O$, décider si $w \in R$). La notion d'effectivité présuppose un encodage fini des régions et des éléments de W. De plus si plusieurs encodages sont possibles pour une même région, on suppose que le test d'égalité est lui-même effectif.

Cas des LCS Dans [BBS06b] nous étudions certains ensembles de configurations pour les LCS, les *régions régulières*, ou simplement *régions*.

Soit $\mathcal{L} = (S, \mathsf{C}, \mathsf{M}, \Delta)$ un LCS.

Définition 1.20 (Région)**.** *$R \subseteq \mathsf{Conf}$ est une* région *si on peut l'écrire sous la forme :*

$$R = \sum_{i\in I}(q_i, R_i^1, \cdots, R_i^{|\mathsf{C}|})$$

où I est un ensemble fini d'indices, chaque q_i est un état de contrôle, et les R_i^j sont des langages réguliers sur l'alphabet M.

Les régions régulières sont closes pour toutes les opérations booléennes et aussi par les opérateurs *Pre* et *Post*.

Nous décrivons une algèbre des régions effective pour les LCS. On rappelle que les régions sont des ensembles réguliers de configurations. Comme opérateurs sur les régions, en plus des opérateurs d'arité zéro ($\emptyset$ et Conf), on inclut l'union, l'intersection, les opérateurs de fermetures ($C_\uparrow$,$C_\downarrow$,$K_\uparrow$ et $K_\downarrow$). Chacun est monotone, effectif et préserve la régularité des régions. On inclut également des opérateurs spécifiques à la vérification : Pre et $\widetilde{Pre}$. Rappelons que pour tout ensemble de configurations A, $\widetilde{Pre}(A) = \neg Pre(\neg A)$. Nous montrons ci-dessous que les deux opérateurs Pre et $\widetilde{Pre}$ sont effectifs et préservent la régularité. Pour cela on se concentre d'abord sur le cas particulier des transitions sans pertes.

$$Pre_{\text{perf}}[p \xrightarrow{c_i?m} r](q, R_p^1, \cdots, R_p^{|\mathsf{C}|}) = \begin{cases} (p, R_p^1, \ldots, R_p^{i-1}, m \bullet R_p^i, R_p^{i+1}, \ldots, R_p^{|\mathsf{C}|}) & \text{si } q = r, \\ \emptyset & \text{sinon.} \end{cases}$$

$$Pre_{\text{perf}}[q \xrightarrow{c_i!m} q'](q, R_p^1, \cdots, R_p^{|\mathsf{C}|}) = \begin{cases} (p, R_p^1, \ldots, R_p^{i-1}, R_p^i m^{-1}, R_p^{i+1}, \ldots, R_p^{|\mathsf{C}|}) & \text{si } q = r, \\ \emptyset & \text{sinon.} \end{cases}$$

$$Pre_{\text{perf}}\Big(\sum_{i\in I}(q_i, R_i^1, \ldots, R_i^{|\mathsf{C}|})\Big) = \sum_{i\in I}\sum_{\delta\in\Delta} Pre_{\text{perf}}[\delta](q_i, R_i^1, \ldots, R_i^{|\mathsf{C}|}).$$

La notation $m \bullet R$ représente la concaténation du message m avec le langage régulier R, et Rm^{-1} est le résiduel à droite de R par m.

Rappelons que dans le cas des LCS, l'ensemble des prédécesseurs d'un ensemble coïncide avec l'ensemble de prédécesseurs de la fermeture vers le haut de cet ensemble (cf. proposition 1.11 page 25). De même, si on considère l'opérateur dual du Pre, *i.e.* le $\widetilde{Pre}$, on peut avant de l'appliquer prendre le noyau fermé vers le bas.

Lemme 1.21. *Soit $A \subseteq \mathsf{Conf}$ un ensemble de configurations.*
- $Pre(A) = Pre(C_\uparrow A)$,
- $\widetilde{Pre}(A) = \widetilde{Pre}(K_\downarrow A)$.

Un μ-calcul Nous étendons à présent l'algèbre des régions avec des points fixes. Soit $\chi = \{X_1, X_2, \cdots\}$ un ensemble dénombrable de variables. On note $L_\mu(W, \sqsubseteq, O)$ (ou succintement L_μ quand à la fois $(W, \sqsubseteq)$ et O sont clairement identifiés par le contexte) l'ensemble des termes sur O avec plus petits et plus grands points fixes donnés par la syntaxe suivante.

$$L_\mu \ni \phi, \psi \ ::= o(\phi_1, \ldots, \phi_k) \mid X \mid \mu X.\phi \mid \nu X.\phi \mid C_\uparrow(\phi) \mid C_\downarrow(\phi) \mid K_\uparrow(\phi) \mid K_\downarrow(\phi)$$

où X est un élément de χ, et o un opérateur d'arité k de O. Notons que les opérateurs $C_\uparrow, C_\downarrow, K_\uparrow, K_\downarrow$ sont déjà présents dans O. On choisit pourtant de les expliciter dans la définition de L_μ à cause de leur rôle particulier.

La sémantique des termes de L_μ est classique :

Définition 1.22 (Environnement)**.** *Un* environnement *est une application $env : \chi \to 2^W$ qui interprète chaque variable $X \in \chi$ comme un sous-ensemble de W.*

Étant donné un environnement env, un terme ϕ de L_μ dénote un sous-ensemble de W, noté $[\![\phi]\!]_{env}$ et défini par induction sur la structure de ϕ. Dans cette définition, pour chaque ϕ, X, env, $\Omega[\phi, X, env]$ est un opérateur unaire définit par $\Omega[\phi, X, env](V) \stackrel{\text{def}}{=} [\![\phi]\!]_{env[X:=V]}$, où $env[X := V]$ est l'environnement qui diffère de env seulement sur X où il vaut V.

$$\begin{aligned} [\![X]\!]_{env} &\stackrel{\text{def}}{=} env(X) & [\![o(\phi_1, \ldots, \phi_k)]\!]_{env} &\stackrel{\text{def}}{=} o([\![\phi_1]\!]_{env}, \ldots, [\![\phi_k]\!]_{env}) \\ [\![C_\uparrow(\phi)]\!]_{env} &\stackrel{\text{def}}{=} C_\uparrow\big([\![\phi]\!]_{env}\big) & [\![C_\downarrow(\phi)]\!]_{env} &\stackrel{\text{def}}{=} C_\downarrow([\![\phi]\!]_{env}) \\ [\![K_\uparrow(\phi)]\!]_{env} &\stackrel{\text{def}}{=} K_\uparrow([\![\phi]\!]_{env}) & [\![K_\downarrow(\phi)]\!]_{env} &\stackrel{\text{def}}{=} K_\downarrow\big([\![\phi]\!]_{env}\big) \\ [\![\mu X.\phi]\!]_{env} &\stackrel{\text{def}}{=} \text{lfp}\big(\Omega[\phi, X, env]\big) & [\![\nu X.\phi]\!]_{env} &\stackrel{\text{def}}{=} \text{gfp}\big(\Omega[\phi, X, env]\big) \end{aligned}$$

Remarquons que $[\![\phi]\!]_{env}$ est indépendant de $env(X)$ si X n'est pas une variable libre dans ϕ. Ainsi, pour une formule close ϕ on notera simplement $[\![\phi]\!]$ plutôt que $[\![\phi]\!]_{env}$.

La sémantique des points fixes dans le treillis complet $(2^W, \subseteq)$ est bien définie puisque, pour tout terme ϕ, pour toute variable X et tout environnement env, l'opérateur $\Omega[\phi, X, env]$ est monotone. De plus, si env et env' sont deux environnements tels que pour toute variable $Y \in \chi$, $env(Y) \subseteq env'(Y)$ alors $\text{lfp}(\Omega[\phi, X, env]) \subseteq \text{lfp}(\Omega[\phi, X, env])$ et $\text{gfp}(\Omega[\phi, X, env]) \subseteq \text{gfp}(\Omega[\phi, X, env])$.

1.3.2 Mu-calcul gardé - Théorème de convergence

Nous définissons à présent la notion centrale de variables gardées vers le haut ou vers le bas.

Définition 1.23 (Termes gardés)**.**
1. *Une variable X est* gardée vers le haut *dans le terme ϕ si chaque occurrence libre de X dans ϕ est sous la portée d'un des opérateurs $C_\uparrow$ ou $K_\uparrow$,* i.e., *apparaît dans un sous-terme de la forme $C_\uparrow(\psi)$ ou $K_\uparrow(\psi)$.*
2. *De façon symétrique, X est* gardée vers le bas *si chacune de ses occurrences libres est sous la portée d'un des opérateurs $C_\downarrow$ ou $K_\downarrow$.*
3. *Un terme ϕ est* gardé *si pour tout sous-terme $\mu X.\psi$, X est gardé vers le haut dans ψ, et pour tout sous-terme $\nu X.\psi$, X est gardé vers le bas dans ψ.*

Définition 1.24. *Étant donnés ϕ, X et env, les* approximations successives *de $[\![\mu X.\phi]\!]_{env}$ sont les $(M_i)_{i\in\mathbb{N}}$, sous-ensembles de W définis par $M_0 = \emptyset$ et $M_{i+1} = [\![\phi]\!]_{env[X:=M_i]}$.*

De façon analogue on définit la suite $(N_i)_{i\in\mathbb{N}}$ des approximations de $\text{gfp}(\Omega[\phi, X, env])$ *par $N_0 = W$ et $N_{i+1} = [\![\phi]\!]_{env[X:=N_i]}$.*

Puisque ϕ est monotone, la suite $(M_i)_{i\in\mathbb{N}}$ est croissante :

$$M_0 \subseteq M_1 \subseteq M_2 \subseteq \cdots \subseteq \text{lfp}(\Omega[\phi, X, env]). \tag{1.1}$$

Pour la suite $(N_i)_{i\in\mathbb{N}}$, la monotonie de ϕ entraîne

$$N_0 \supseteq N_1 \supseteq N_2 \supseteq \cdots \supseteq \text{gfp}(\Omega[\phi, X, env]). \tag{1.2}$$

Lemme 1.25 (Convergence finie des approximations)**.** *Soient X une variable et ϕ un terme de L_μ. Si X est gardé vers le haut dans ϕ alors il existe un indice $k \in \mathbb{N}$ tel que*

$$[\![\mu X.\phi]\!]_{env} = M_k = M_{k+1} = M_{k+2} = \cdots \tag{1.3}$$

De façon analogue, si X est gardé vers le bas dans ϕ alors il existe $k' \in \mathbb{N}$ tel que

$$[\![\nu X.\phi]\!]_{env} = N_{k'} = N_{k'+1} = N_{k'+2} = \cdots \tag{1.4}$$

Preuve : Il suffit de prouver la première partie du lemme, la seconde étant obtenue par dualité.

Soit ϕ un terme de L_μ. On note $\psi_1, \cdots, \psi_m$ les sous-termes maximaux de ϕ qui sont sous la portée immédiate d'un opérateur $C_\uparrow$ ou $K_\uparrow$. Le terme ϕ peut alors être écrit sous la forme

$$\phi \;\equiv\; \Phi(\Uparrow \psi_1, \ldots, \Uparrow \psi_m)$$

où $\Uparrow$ remplace soit $C_\uparrow$ soit $K_\uparrow$. Dans les deux cas, et pour tout environnement env', l'ensemble $[\![\Uparrow \psi_i]\!]_{env'}$ est fermé vers le haut. On suppose que le contexte $\Psi(Y_1, \cdots, Y_m)$ utilise des nouvelles variables $Y_1, \cdots, Y_m$ qui sont substituées par les $(\Uparrow \psi_i)_i$.

Si $V_1, \cdots, V_m$ sont des sous-ensembles de W, on notera simplement $[\![\Psi]\!](V_1, \cdots, V_m)$ à la place de $[\![\Psi]\!]_{env[Y_1:=V_1,\cdots,Y_m:=V_m]}$. Par hypothèse, X est gardé vers le haut dans ψ, donc ses occurrences sont toutes dans les ψ_i et aucune ne se trouve dans Ψ. Ainsi

$$\begin{aligned} M_{i+1} = [\![\phi]\!]_{env[X:=M_i]} &= [\![\Phi]\!]([\![\Uparrow \psi_1]\!]_{env[X:=M_i]}, \ldots, [\![\Uparrow \psi_m]\!]_{env[X:=M_i]}) \\ &= [\![\Phi]\!](L_{i,1}, \ldots, L_{i,m}) \end{aligned}$$

en notant $L_{i,j}$ l'ensemble $[\![\Uparrow \psi_j]\!]_{env[X:=M_i]}$.

De l'équation (1.1) qui exprime en particulier la croissance des $(M_i)_i$, on tire $L_{0,j} \subseteq L_{1,j} \subseteq L_{2,j} \subseteq \cdots$. D'autre part le résultat de l'application d'un des opérateurs $C_\uparrow$ ou $K_\uparrow$ est un ensemble fermé vers le haut. Les $L_{i,j}$ forment donc à indice j fixé une suite croissante d'ensembles fermés vers le haut. On utilise alors la Proposition 1.9 (page 24) qui donne pour chaque $j = 1, \cdots, m$ un entier k_j à partir duquel la suite $(L_{i,j})_i$ se stabilise. Alors en choisissant $K = \max(k_1, \ldots, k_j)$ on obtient pour tout $i \geq K$

$$M_{i+1} = [\![\Phi]\!](L_{i,1}, \ldots, L_{i,m}) = [\![\Phi]\!](L_{k_1,1}, \ldots, L_{k_m,m}) = [\![\Phi]\!](L_{K,1}, \ldots, L_{K,m}) = M_{K+1}.$$

Ainsi $\bigcup_{i\in\mathbb{N}} M_i = M_{K+1} = M_{K+2}$ et M_{K+1} est un point fixe de $\Omega[\phi, X, env]$. C'est même le plus petit point fixe grâce à l'équation (1.1). Choisir $k = K + 1$ permet donc de conclure. $\square$

À présent, nous présentons le résultat principal sur L_μ. Les sous-ensembles définis par des termes de L_μ sont des régions, et peuvent être calculés effectivement dès que l'algèbre des régions est effective.

On parle d'*environnement de régions* pour désigner un environnement $env : \chi \to \mathcal{R}$ qui associe à chaque variable une région. Si env est un environnement de régions, et si le terme ϕ ne possède pas de sous-terme $\mu X.$ ou $\nu Y.$ (toutes les variables sont libres), alors $[\![\phi]\!]_{env}$ est une région.

Théorème 1.26. *Soit ϕ un terme de L_μ et env un environnement de régions. Si ϕ est gardé, alors $[\![\phi]\!]_{env}$ est une région. De plus, si l'algèbre des régions est effective, $[\![\phi]\!]_{env}$ peut être calculé de façon effective à partir de ϕ et env.*

Preuve : La preuve se fait par induction sur la structure des termes de L_μ.

Si ϕ est un opérateur 0-aire (*i.e.*, une constante), le résultat repose sur la définition de l'algèbre des régions.

Supposons $\phi = o(\phi_1, \cdots, \phi_k)$ où chaque $[\![\phi_i]\!]_{env}$ est une région (effective) par hypothèse d'induction. Alors $[\![\phi]\!]_{env}$ est également une région (effective) par définition de la sémantique de L_μ. Cet argument s'applique en particulier aux opérateurs d'arité zéro et aux opérateurs unaires que l'on met en exergue : $C_\uparrow$, $C_\downarrow$, $K_\uparrow$ et $K_\downarrow$.

Soit $\phi = \mu X.\psi$. On commence par montrer par récurrence que chaque approximation $M_0, M_1, M_2, \cdots$ de $[\![\phi]\!]_{env}$ est une région. $M_0 = \emptyset$ est bien une région, et si M_i est une région, alors M_{i+1} en est une également puisque $env' = env[x := M_i]$ est un environnement de régions et que par hypothèse d'induction $[\![\psi]\!]_{env'}$ est une région si env' est un environnement de régions. Lorsque $\mathcal{R}_O$ est effective, on peut calculer effectivement les ensembles M_i, et donc détecter à quel moment $M_k = M_{k+1}$ car l'égalité des régions est décidable. Alors $[\![\phi]\!]_{env} = M_k$ peut également être calculé de façon effective.

Enfin, le cas $\phi = \nu X.\psi$ est dual. $\square$

Exemple 1.27 (Une algèbre des régions pour les langages réguliers). *Considérons $W = \Sigma^*$ l'ensemble des mots fini sur un alphabet fini Σ. La relation de sous-mot $\sqsubseteq$ est un bel ordre (cf. Théorème 1.5 page 24). Les langages réguliers sur Σ constituent un choix naturel pour les régions. De plus les opérateurs de clôture ($C_\uparrow$ et $C_\downarrow$) préservent la régularité et peuvent être implémentés. En effet, à partir d'un automate fini pour $R \subseteq \Sigma^*$ régulier, on obtient un automate fini pour $C_\uparrow R$ en ajoutant des boucles $q \xrightarrow{a} q$ pour tous les états q, et toutes les lettres $a \in \Sigma$. Un automate fini pour $C_\downarrow R$ est obtenu en ajoutant des ε-transitions $q \xrightarrow{\varepsilon} q'$ à chaque fois qu'une transition $q \xrightarrow{a} q'$ (avec $a \in \Sigma$) existe.*

D'autres opérateurs sont naturellement ajoutés à l'algèbre des régions, comme l'union et l'intersection. En fait, toute opération sur les langages qui est monotone, préserve la régularité, et peut être implémentée, peut faire partie de l'algèbre des régions. C'est par exemple le cas de la concaténation (notée $R \bullet R'$), de l'étoile de Kleene (notée R^*), des résiduels à gauche ou à droite (notés $R^{-1}R'$ et $R'R^{-1}$), du mélange (le shuffle *noté $R \sqcup\!\sqcup R'$), du miroir (noté $\overleftarrow{R}$), de la conjugaison (notée $\tilde{R}$), des images par homomorphisme et homomorphisme inverse, et beaucoup d'autres [Per90].*

La complémentation, n'est pas autorisée puisque cet opérateur n'est pas monotone, cependant le complément peut être utilisé avec la restriction que les variable liées soient sous la portée d'un nombre pair d'opérateurs de complément. En effet, si o est monotone, effectif et préserve la régularité, alors son dual, noté $\tilde{o}$ l'est aussi.

Une application du Théorème 1.26 est que pour tous langages réguliers R_1 et R_2, le langage défini par $\mu X.\nu Y.\big(K_\uparrow\big[R_1 \parallel (X^* \cap C_\downarrow(Y^{-1}\overleftarrow{X} \cap X^{-1}R_2))\big]\big)$ est régulier et on peut en calculer une représentation finie à partir de R_1 et R_2.

Considérons à présent trois problèmes de décision sur les algèbres de régions : le model checking, la satisfaisabilité et l'universalité.

> **Model checking**
> **Instance :** Un élément $w \in W$ et un terme $\phi \in L_\mu$ clos et gardé.
> **Question :** $w \in [\![\phi]\!]$?
>
> **Satisfaisabilité**
> **Instance :** Un terme $\phi \in L_\mu$ clos et gardé.
> **Question :** $[\![\phi]\!] \neq \emptyset$?
>
> **Universalité**
> **Instance :** Un terme $\phi \in L_\mu$ clos et gardé.
> **Question :** $[\![\phi]\!] = W$?

Corollaire 1.28. *Les problèmes de model-checking, satisfaisabilité et universalité sont décidables dans les algèbres de régions monotones et effectives.*

1.3.3 Vérification des LCS

Le Théorème 1.26 possède de nombreuses applications dans le domaine de la vérification des LCS et d'autres familles de systèmes bien structurés. Nous donnons ici quelques exemples d'applications. Le Théorème 1.26 sera également très fréquemment utilisé dans les preuves de décidabilité de la troisième partie.

Accessibilité Dans tout système de transitions, l'ensemble des prédécesseurs de A en un nombre arbitraire d'étapes s'exprime de la façon suivante :

$$Pre^*(A) = \mu X.A \vee Pre(X).$$

Dans le cas particulier des LCS, l'ensemble des prédécesseurs en une étape d'un ensemble quelconque de configurations coïncide avec l'ensemble des prédécesseurs du fermé vers le haut. Ceci est dû aux pertes de messages qui peuvent survenir après que les transitions parfaites sont tirées.

$$Pre(A) = Pre(C_\uparrow A).$$

Notons que cette propriété est fortement liée à la sémantique que nous avons choisie pour les pertes. Dans le cas où les pertes peuvent se produire à la fois avant et après les actions parfaites (comme dans [AJ96b]), on aurait l'égalité $Pre(A) = C_\uparrow Pre(C_\uparrow A)$. Enfin, si les pertes avaient lieu uniquement avant les actions l'équation deviendrait $Pre(A) = C_\uparrow Pre(A)$. Néanmoins, ces modifications n'altéreraient pas le caractère gardé des termes utilisant Pre car, à chaque fois, l'ensemble A est sous la portée de l'opérateur $C_\uparrow$, et il est donc donc gardé vers le haut.

Lemme 1.29 (Prédécesseurs)**.** *Soit $\mathcal{L}$ un LCS et A un ensemble quelconque de configurations. Alors,*

$$Pre^*(A) = \mu X.A \vee Pre(C_\uparrow X)$$

Corollaire 1.30. *Étant donnés $\mathcal{L}$ un LCS et $R \subseteq \mathsf{Conf}$ une région, l'ensemble $Pre^*(R)$ des prédécesseurs de R est une région et peut être calculé effectivement.*

Propriétés de sûreté Au delà des propriétés d'accessibilité, nous nous intéressons maintenant aux propriétés de sûreté et montrons que l'on peut les traiter grâce au Théorème 1.26. Une propriété de sûreté s'exprime dans la logique CTL sous la forme : $\forall(V_1 \mathsf{R} V_2)$ en utilisant l'opérateur «Release» R . Rappelons que R est le dual de la modalité *Until* : un chemin π satisfait $\forall(V_1 \mathsf{R} V_2)$ si et seulement si le long de π, $\mathcal{V}_2$ est toujours vrai jusqu'à ce que V_1 soit peut-être atteint.

Grâce au Lemme 1.21 (page 30), on écrit $[\![\forall(V_1 \mathsf{R} V_2)]\!]$, l'ensemble des configurations pour lesquelles la propriété de sûreté est vraie sous la forme suivante :

$$\begin{aligned}
&[\![\forall(V_1 \mathsf{R} V_2)]\!] \\
&= \nu X.\big(V_2 \cap (\widetilde{Pre}(X) \cup V_1)\big)\big) \text{ par [EC80]} \\
&= \nu X.\big(V_2 \cap \big(\widetilde{Pre}(K_\downarrow(X)) \cup V_1\big)\big) \text{ grâce au Lemme 1.21.}
\end{aligned}$$

Cette dernière équation permet d'écrire $[\![\forall(V_1 \mathsf{R} V_2)]\!]$ comme un terme gardé de L_μ. On en déduit :

Corollaire 1.31. *Pour toutes régions $V_1, V_2 \subseteq \mathsf{Conf}$, $[\![\forall(V_1 \mathsf{R} V_2)]\!]$ est une région et peut être calculé effectivement.*

Au delà de la sûreté On peut écrire dans L_μ des propriétés d'inévitabilité ou d'accessibilité répétée. Pour l'inévitabilité par exemple :

$$[\![\forall\Diamond V]\!] = \mu X.\big(V \cup (Pre(\mathsf{Conf}) \cap \widetilde{Pre}(X))\big)$$

Ce terme n'est pas gardé, et le Lemme 1.21 (page 30) ne permet pas de le transformer en un terme gardé. Ceci n'est pas surprenant puisque, pour les LCS, même si on peut décider si une configuration donnée satisfait $\forall\Diamond V$, l'ensemble de configurations $[\![\forall\Diamond V]\!]$ n'est pas calculable en général [May03].

Pour les propriété d'accessibilité répétée, on fait un constat similaire. Le terme suivant

$$[\![\exists\Box\Diamond V]\!] = \nu X.\big(\mu Y.((V \cup Pre(Y)) \cap Pre(X))\big)$$

n'est pas gardé, mais $[\![\exists\Box\Diamond V]\!]$ n'est pas calculable en général. Et même, savoir si une configuration σ satisfait $\exists\Box\Diamond V$ est indécidable [AJ96a].

Remarque 1.32 (Généralisation aux LCS étendus)**.** *Tous les résultats de calculabilité présentés ci-dessus s'étendent de façon limpide aux LCS étendus. En effet, il suffit de remarquer que les propriétés utilisées (à savoir celles du Lemme 1.21) sont toujours valables et que les opérateurs* Pre *et* $\widetilde{Pre}$ *sont encore effectifs.*

1.3.4 Historique des représentations symboliques

Plusieurs représentations ont été proposées pour les ensembles de configurations dans le cadre du regular model-checking des automates communicants. Nous terminons ce chapitre par un aperçu de ces diverses représentations.

Cas des canaux parfaits Dans un premier temps, et pour les automates communicants à canaux parfaits, Finkel et Marcé [FM96] proposent une procédure d'analyse symbolique par le biais d'expressions régulières d'un certain type. Cependant l'ensemble d'accessibilité calculé par cette procédure n'est pas toujours exact.

Boigelot *et al.* [BG99, BGWW97] utilisent des automates finis, les QDD, pour représenter les contenus des canaux. Les QDD permettent de représenter des ensembles reconnaissables de suites de vecteurs. Ces structures ne sont pas stables par l'itération des boucles dès que l'automate communicant possède deux canaux. Pour calculer et représenter l'effet d'une boucle dans le cas des canaux parfaits, il est nécessaire de passer aux CQDD qui sont des QDD contraints [BH97].

Expressions régulières simples Pour les automates communicants à pertes, les relations sur le nombre d'occurrences des messages dans les canaux, qui sont exprimées par les CQDD ne sont plus valables, à cause des pertes possibles. C'est pourquoi Abdulla *et al.* ont défini les SRE (*simple regular expressions*) qui sont adaptées aux LCS et permettent de calculer l'effet de boucles bien plus facilement que dans le cas de canaux parfaits. Les SRE sont des expressions régulières particulières qui sont utilisées pour le calcul en avant (calcul des successeurs d'un ensemble initial). Elles présentent l'avantage de posséder une forme canonique, qui est de plus calculable en temps quadratique.

Définissons les SRE sur l'alphabet de messages M.

Définition 1.33 (SRE)**.** *Une* expression atomique *est une expression régulière de la forme*
- $(m + \varepsilon)$ *où* $m \in \mathsf{M}$*, ou*
- $(m_1 + \cdots + m_l)^*$ *où* $m_1, \cdots, m_l \in \mathsf{M}$.

Un produit *p est la concaténation (éventuellement vide) $e_1 \bullet \cdots \bullet e_n$ d'expressions atomiques $e_1, \cdots, e_n$. On note ε le produit vide, expression régulière qui dénote le singleton $\{\varepsilon\}$.*

Une SRE *(expression régulière simple) r est de la forme $p_1 + \cdots + p_m$ où $p_1 \cdots p_m$ sont des produits. On note $\emptyset$ la somme vide qui dénote le langage vide.*

Des méthodes sont données dans [BH97] pour tester l'inclusion de SRE en temps quadratique, calculer la SRE obtenue par l'effet d'une transition, ou un nombre d'itérations quelconques d'une boucle dans un LCS. Ces techniques permettent alors de construire un algorithme d'exploration de l'ensemble des états accessibles. Les SRE et leur algorithmes dédiés, ont été implémentées dans l'outil TReX [ABS01].

Contraintes Abdulla, Bouajjani et d'Orso ont introduit plusieurs types de *contraintes* (cf. [ABd03]). Les briques de base qu'ils définissent sont les contraintes fermées vers le haut qui représentent des ensembles fermés vers le haut. Les contraintes fermées vers le haut étendues sont alors de la forme $x \bullet \phi$ où ϕ est une contrainte fermée vers le haut, et x un mot sur l'alphabet des messages M. Ces dernières permettent de spécifier le début des contenus des canaux. Par exemple $m_1 m_2 \uparrow (s, \uparrow \varepsilon)$ dénote l'ensemble des configurations ayant s comme état de contrôle et dont le contenu de canal commence par $m_1 m_2$. Pour ce type de représentations, l'intersection est calculable de façon effective, l'ensemble des prédécesseurs également, et l'appartenance d'une configuration est décidable. Ceci leur permet de résoudre des jeux d'accessibilité (ou d'invariant) sur des LCS où l'un des joueurs contrôle à la fois les actions qu'il tire, et les pertes qui suivent ces actions.

Ensembles contraints Kučera et Schnoebelen [KS06] ont défini des *ensembles contraints*, qui cette fois-ci permettent une étude d'accessibilité en arrière, c'est-à-dire en calculant l'ensemble des prédécesseurs d'un état final.

Leur travail repose sur une sémantique des LCS un peu différente de celle adoptée dans cette thèse et que nous avons déjà évoquée : les pertes de messages peuvent se produire à la fois avant et après les transitions parfaites.

Définition 1.34. *Un* ensemble restreint *est dénoté par une expression ρ de la forme $\uparrow\sigma - \uparrow\tau_1 - \cdots - \uparrow\tau_n$ où σ et $\tau_1 \cdots \tau_n$ sont des configurations. L'ensemble restreint ρ représente les configurations qui sont dans $\uparrow\sigma$ sans être dans une «restriction» τ_i.*

Un ensemble contraint *est une union finie d'ensembles restreints : $\gamma = \rho_1 + \cdots + \rho_m$.*

Du point de vu expressivité, les ensembles contraints représentent la clôture booléenne des SRE, qui ne sont pas stables par complémentation. Les ensembles contraints sont bien adaptés à l'étude en arrière des LCS puisqu'en plus d'être clos par les opérations booléennes de façon effective, ils sont aussi clos par prédécesseurs. Étant donné un ensemble contraint γ on peut calculer γ' tel que $\gamma' = Pre(\gamma)$ (ou plus précisément, l'ensemble dénoté par γ' est constitué des prédécesseurs de l'ensemble dénoté par γ). On peut même calculer les ensembles de configurations satisfaisant une formule du type $\exists A\ Until\ B$ ou $\exists\ \mathsf{X}\ B$, ce qui permet de vérifier des propriétés d'accessibilité contrainte.

Remarque 1.35. *Si on fait le choix de la sémantique opérationnelle des LCS où les pertes n'ont lieu qu'après les transitions parfaites, les ensembles contraints ne sont plus stables par l'opérateur Pre. Cependant, on peut modifier légèrement leur définition, et établir de nouveaux algorithmes pour traiter cette sémantique-là. Nous le verrons en détail dans la quatrième partie.*

Deuxième partie

Lossy Channel Systems probabilistes

Chapitre 2

LCS probabilistes

Sommaire

La sémantique non déterministe des LCS telle que vue au Chapitre 1 est trop pessimiste, en particulier par rapport aux propriétés de vivacité. En effet, rien n'assure qu'un message, même retransmis un grand nombre de fois, finira par ne pas être perdu et pourra être lu par un récepteur. Une sémantique probabiliste pour les pertes permet de pallier cet inconvénient, puisqu'elle entraîne une équité forte sur les pertes de messages : si un message est envoyé sur un canal infiniment souvent, presque sûrement il finira par être transmis.

Une autre motivation pour l'introduction d'une sémantique probabiliste est la possibilité dans ce formalisme de considérer des questions quantitatives. Lorsque la probabilité de perte d'un message est prise en compte dans un modèle, une question naturelle est la suivante : « La probabilité qu'une erreur survienne est-elle basse ? » Ou de façon similaire : « La probabilité que le système fonctionne correctement est-elle supérieure à 0.9 ? ».

2.1 La sémantique markovienne des LCS

Après avoir fait quelques rapides rappels relatifs aux probabilités, on présente une sémantique des LCS sous forme de chaînes de Markov. Pour cela, on définit un taux de perte, et on munit les actions de poids. Ce modèle a été présenté de façon indépendante dans [AR03] et [BS03].

2.1.1 Chaînes de Markov

Nous ne considérerons que des chaînes de Markov à temps discret (qui représentent les étapes), à espace d'état dénombrable (ce sont les configurations) et homogènes (la probabilité de passer d'une configuration à une autre n'évolue pas au cours du temps). On donne donc la définition de chaîne de Markov dans ce cadre particulier. Aussi, dans la suite, on parlera simplement de chaîne de Markov pour « chaîne de Markov à temps discret, espace d'état dénombrable, et homogène ».

Pour une définition générale des chaînes de Markov nous renvoyons à la lecture d'ouvrages de probabilités (comme par exemple [Ouv98, Ouv00]). D'autre part, nous signalons l'existence d'un article qui introduit la théorie de la mesure et les probabilités pour les informaticiens [Pan01].

Définition 2.1. *Une* chaîne de Markov *est un couple* $\mathcal{M} = (E, P)$ *où* E *est un ensemble dénombrable d'états ;* $P : E \times E \to [0, 1]$*, la* probabilité de transition *qui vérifie* $\forall x \in E,\ \sum_{y \in E} P(x, y) = 1$.

La fonction de probabilité de transition peut être vue comme une *matrice de transition* en ordonnant les états de E :

$$\mathbf{P} \stackrel{\text{def}}{=} (P(x, y))_{x,y \in E}.$$

On dit que y est un *successeur* de x si $\mathbf{P}(x, y) > 0$.

Remarque 2.2. *La définition 2.1 implique en particulier que tout état possède au moins un successeur. Si dans le système à modéliser ce n'est pas le cas, on augmente les états sans successeurs d'une boucle de probabilité* 1.

Système de transition sous-jacent Une chaîne de Markov homogène à espace d'état fini peut être représentée par un graphe orienté et étiqueté. Les nœuds sont les états de $\mathcal{M}$ et les étiquettes sont les probabilités données par $\mathbf{P}$. Chaque étiquette est un réel compris entre 0 et 1 et pour tout état $i \in E$, la somme des étiquettes des arêtes sortantes vaut 1.

La chaîne de Markov $\mathcal{M}$ induit un *système de transition sous-jacent* $\mathsf{ST}_{\mathcal{M}} \stackrel{\text{def}}{=} (E, \to)$, ayant pour ensemble d'états E et tel que la relation de transition $\to$ vérifie : $s \to t$ si et seulement si $\mathbf{P}(s, t) > 0$. Dans la suite on s'autorise à utiliser des notions ayant trait au système de transition en parlant de la chaîne de Markov. Par exemple, les composantes fortement connexes finales[1] de $\mathcal{M}$ sont en fait celles de son système de transition sous-jacent.

Prenons comme exemple la chaîne de Markov $\mathcal{M} = (E, \mathbf{P})$ avec E et la matrice de transitions $\mathbf{P}$ donnés ci-dessous :

$$E = \{s_0, s_1, s_2, s_3, s_4, s_5\} \qquad \mathbf{P} = \begin{pmatrix} 0 & 1/4 & 1/4 & 1/2 & 0 & 0 \\ 1/4 & 3/4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 2/3 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 2/3 & 1/3 \end{pmatrix} \tag{2.1}$$

Le système de transition étiqueté sous-jacent est donné Figure 2.1.

[1]cf. Définition 2.16 page 47.

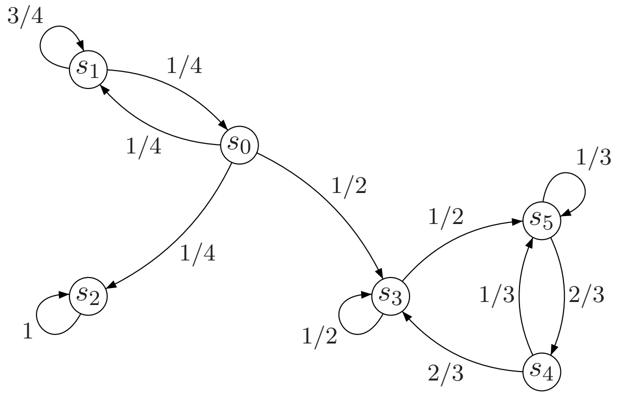

FIG. 2.1 – Représentation graphique de la chaîne de Markov (2.1)

Mesure d'ensembles d'exécutions Une exécution de $\mathcal{M} = (E, \mathbf{P})$ à partir de x_0 est une suite infinie d'états $x_0, x_1, \ldots$ de E. Si on fixe un état initial x_0, une chaîne de Markov $\mathcal{M} = (E, \mathbf{P})$ est munie naturellement d'une mesure sur l'ensemble de ses exécutions. Précisément, soit x_0 un état de $\mathcal{M}$; l'espace de probabilités $\langle \Omega, \Delta, \mathbb{P} \rangle$ sur les exécutions partant de x_0 est défini comme suit :

- $\Omega = x_0 E^\omega$ est l'ensemble des exécutions infinies partant de x_0,
- Δ est la σ-algèbre engendrée par les cylindres $D_\pi = \pi E^\omega$ pour tout chemin fini $\pi \in x_0 E^*$,
- $\mathbb{P}$ est la mesure définie par $\mathbb{P}(D_\pi) \stackrel{\text{def}}{=} \prod_{0 \leq i < n} \mathbf{P}(x_i, x_{i+1})$ si $\pi : x_0 \cdots x_n$. Elle est étendue à tous les ensembles mesurables.

Sur l'exemple de la Figure 2.1, et pour le chemin fini $\pi = s_0 s_1 s_0$ on a $\mathbb{P}(D_\pi) = \frac{1}{4} . \frac{1}{4} = \frac{1}{16}$ et le cylindre D_π est représenté Figure 2.2.

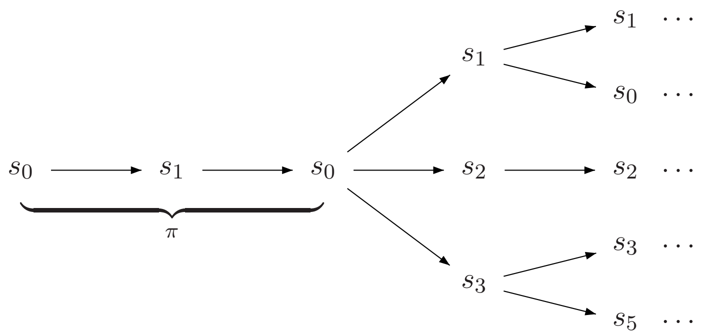

FIG. 2.2 – Ensemble des chemins du cylindre $s_0 s_1 s_0$

On peut alors parler de la mesure d'ensemble de chemins, sous réserve que cet ensemble soit mesurable. Un résultat dû à Vardi [Var85] montre que l'ensemble des chemins qui satisfont une propriété exprimée dans la logique LTL, ou de façon équivalente par un automate de mots infinis, est un ensemble mesurable.

Proposition 2.3 ([Var85])**.** *Soit $\mathcal{M} = (E, \mathbf{P})$ une chaîne de Markov et $\mathcal{B}$ un automate de Büchi sur l'alphabet E. Alors l'ensemble $\mathcal{M}(\mathcal{B})$ des exécutions $\pi : x_1 x_2 \cdots$ de $\mathcal{M}$ telles que $x_1 x_2 \cdots$ est accepté par $\mathcal{B}$ est mesurable.*

La traduction de formules LTL en automates des Büchi permet alors de conclure que les ensembles de chemins d'une chaîne de Markov définis par une formule LTL sont mesurables. On note $\Pr(x \models \phi)$ pour la mesure de l'ensemble des chemins issus de x qui satisfont la propriété ϕ. Traditionnellement, on dit d'un événement de mesure 1 qu'il est presque sûr, ou qu'il arrive presque sûrement. Ainsi, si $\Pr(x \models \phi) = 1$, on dira que la propriété ϕ est vraie presque sûrement à partir de x.

Lemme 2.4. *Soit $\mathcal{M}$ une chaîne de Markov et x_1, x_2 deux de ses états. Si $x_1 \xrightarrow{*} x_2$, alors pour tout état x, $\Pr(x \models \Box\Diamond x_1 \wedge \Box\Diamond x_2) = \Pr(x \models \Box\Diamond x_1)$.*

Idée de la preuve. Comme x_2 est atteignable depuis x_1, chaque fois que x_1 est visité, il y a une probabilité non nulle d'atteindre x_2, en un nombre fini d'étapes. Alors si x_1 est visité infiniment souvent, x_2 sera visité presque sûrement, et en itérant ce raisonnement, x_2 sera même visité infiniment souvent avec probabilité 1. □

2.1.2 Les LCS probabilistes ou PLCS

Nous proposons à présent une sémantique en terme de chaîne de Markov pour les Lossy Channel Systems. Cette vision est particulièrement adaptée dans le cas où à la fois les actions et les pertes suivent une loi de probabilité. Le modèle obtenu, qui en plus du LCS intègre un taux de perte des messages et un poids pour les actions, est appelé LCS probabiliste, abrégé en PLCS.

Baier et Engelen [BE99] ont introduit une sémantique markovienne pour les PLCS pour laquelle il y a une probabilité τ fixée que la prochaine étape soit une perte. Pour ce modèle, appelé *modèle à fautes globales* si $\tau > \frac{1}{2}$, la vérification qualitative est décidable : on peut décider si une formule de la logique du temps linéaire est satisfaite presque sûrement. Cependant, ce résultat ne tient plus lorsque le taux de perte est faible. La décidabilité est obtenue pour un taux supérieur à 0.5, une valeur discutable pour certains systèmes. Pour un modèle légèrement différent [ABPJ00] montrent que la vérification est indécidable si τ n'est pas assez grand. Le fait que la décidabilité dépende du taux de pertes, qui est fixé de façon souvent arbitraire, est un inconvénient. D'autre part, il est naturel de penser que la probabilité de perdre un message est plus grande si le nombre de messages en transit augmente.

Pour ces raisons, nous introduisons un modèle plus réaliste, appelé *modèle à fautes locales* : tout message a une probabilité $\tau > 0$ d'être perdu à chaque étape, indépendamment des autres messages présents dans les canaux. Une conséquence est que plus un canal contient de messages, plus il y a des chances qu'un message soit perdu.

Rappelons que nous avons fait le choix dans cette thèse d'une sémantique des LCS (et donc des PLCS) où les pertes ont lieu après les actions du système.

Définition 2.5. *Un* LCS probabiliste (PLCS) *$\mathcal{P}$ est un n-uplet $\langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w\rangle$ où $\mathcal{L} \stackrel{\text{def}}{=} \langle S, C, \mathsf{M}, \Delta\rangle$ est un LCS, $\lambda \in]0,1[$ est le taux de perte et w une fonction de poids qui associe à chaque règle $\delta \in \Delta$ un entier positif $w(\delta) \in \mathbb{N} \setminus \{0\}$.*

Exemple 2.6. *La Figure 2.3 donne un exemple de PLCS. Les canaux c_1 et c_2 ne sont pas représentés. Les arêtes sont étiquetées à la fois par l'opération des règles de transition et le poids attribué à la règle par la fonction de poids w.*

$\lambda = \mathbf{.01}$

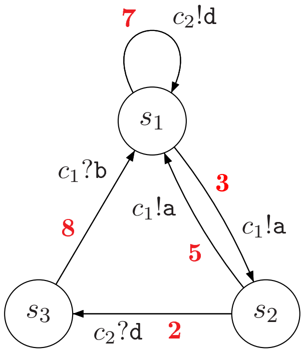

FIG. 2.3 – Un exemple de PLCS

La chaîne de Markov $\mathcal{M}_\mathcal{P}$ À partir d'un PLCS $\mathcal{P}$, on construit une chaîne de Markov en attribuant des probabilités aux transitions du système de transition sous-jacent : cette chaîne de Markov $\mathcal{M}_\mathcal{P}$ constitue la sémantique de $\mathcal{P}$. Dans une configuration donnée, le choix entre les règles se fait de façon probabiliste en tenant compte du poids de chacune des règles tirables. De plus, après chaque action, les messages peuvent être perdus, chacun avec une probabilité λ, indépendamment des autres. Ainsi la probabilité d'atteindre en une étape $(s_2, \mathbf{w}_2)$ à partir de $(s_1, \mathbf{w}_1)$ est égale à la probabilité d'atteindre un certain $(s_3, \mathbf{w}_3)$ par une action du LCS multipliée par la probabilité d'atteindre $(s_2, \mathbf{w}_2)$ à partir de $(s_3, \mathbf{w}_3)$ par des pertes de messages.

Nous définissons ci-dessous plus formellement les probabilités de la chaîne de Markov $\mathcal{M}_\mathcal{P}$.

On s'intéresse d'abord aux probabilités de changer de configuration par des pertes de messages. Soient x, y des mots sur l'alphabet M, représentant le contenu d'un canal. On définit $\#(x, y)$ comme la taille de l'ensemble :

$$\{(i_1, \ldots, i_n) | i_1 < \ldots < i_n \text{ et } x = y(i_1) \ldots y(i_n)\}$$

En d'autre termes, $\#(x, y)$ est le nombre de façons d'obtenir x en ignorant des lettres de y.

Exemple 2.7.
- *higman* $\sqsubseteq$ *highmountain et* $\#(higman, highmountain) = 1$,
- *aba* $\sqsubseteq$ *abracadabra et* $\#(aba, abracadabra) = 8$,
- $x \not\sqsubseteq y$ *ssi* $\#(x, y) = 0$.

On définit alors :

$$P_L(x, y) \stackrel{\text{def}}{=} \#(y, x)\lambda^{|x|-|y|}\dot{(}1-\lambda)^{|y|}$$

qui est la probabilité que x devienne y en perdant des symboles, si chaque symbole peut être perdu (indépendamment des autres) avec probabilité λ.

Si $|x| \geq k$, $\binom{|x|}{k}$ dénote le nombre de façons de choisir k positions parmi les lettres de x, donc correspond au nombre de sous-mots de longueur k de x. Lors de ce décompte, un sous-mot y apparaît $\#(y, x)$ fois. Ainsi $\sum_{y \in M^k} \#(y, x) = \binom{|x|}{k}$.

Vérifions que $\sum_{y \in \mathsf{M}^*} P_L(x, y) = 1$ pour tout $x \in \mathsf{M}^*$:

$$\begin{aligned}\sum_{y \in \mathsf{M}^*} P_L(x, y) &= \sum_{y \in \mathsf{M}^*} \#(y, x) \cdot \lambda^{|x|-|y|} \cdot (1-\lambda)^{|y|} \\ &= \sum_{k \in \mathbb{N}} \lambda^{|x|-k} \cdot (1-\lambda)^k \cdot \sum_{y \in \mathsf{M}^k} \#(y, x).\end{aligned}$$

En prenant en compte que $\sum_{y \in M^k} \#(y,x) = \binom{|x|}{k}$ on obtient donc :

$$\begin{aligned}
\sum_{y \in \mathsf{M}^*} P_L(x,y) &= \sum_{k \in \mathbb{N}} \lambda^{|x|-k} \cdot (1-\lambda)^k \cdot \binom{|x|}{k} \\
&= [\lambda + (1-\lambda)]^{|x|} && \text{grâce à la formule du binôme} \\
&= 1
\end{aligned}$$

La fonction $y \to P_L(x,y)$ définie pour les mots sur l'alphabet M vérifie bien que la somme des probabilités $P_L(x,y)$ pour un x fixé vaut 1, ce qui en fait une loi de probabilité sur M^*. Cette loi de probabilité s'étend de façon naturelle aux contenus de canaux $\mathbf{w} \in {\mathsf{M}^*}^{\mathsf{C}}$, en faisant le produit des probabilités pour chacun des canaux. Plus précisément, si $\mathbf{w}_1$ et $\mathbf{w}_2$ sont des contenus de canaux (*i.e.*, des applications de C dans M^*), on définit $P_L(\mathbf{w}_1, \mathbf{w}_2) \stackrel{\text{def}}{=} \prod_{c \in \mathsf{C}} P_L(\mathbf{w}_1(c), \mathbf{w}_2(c))$.

Pour deux configurations $(s_1, \mathbf{w}_1)$ et $(s_2, \mathbf{w}_2)$ on pose

$$P_L((s_1,\mathbf{w}_1),(s_2,\mathbf{w}_2)) \stackrel{\text{def}}{=} \begin{cases} P_L(\mathbf{w}_1,\mathbf{w}_2) & \text{si } s_1 = s_2, \\ 0 & \text{sinon.} \end{cases} \tag{2.2}$$

Ainsi, pour toute configuration $\sigma \in \mathsf{Conf}$,

$$\sum_{\tau \in \mathsf{Conf}} P_L(\sigma, \tau) = 1. \tag{2.3}$$

Pour prendre en compte le poids des différentes transitions tirables dans une configuration précise, on introduit :

$$w(\sigma) \stackrel{\text{def}}{=} \sum_{\delta \in \Delta(\sigma)} w(\delta)$$

la somme des poids de toutes les transitions tirables dans la configuration $(s, \mathbf{w})$.

On peut alors donner précisément la chaîne de Markov $\mathcal{M}_\mathcal{P} \stackrel{\text{def}}{=} (\mathsf{Conf}, \mathbf{P})$ induite par le PLCS $\mathcal{P}$. Les états sont les configurations du LCS sous-jacent à $\mathcal{P}$, et la matrice de transition $\mathbf{P}$ est définie par :

$$\mathbf{P}(\sigma,\tau) \stackrel{\text{def}}{=} \sum_{\delta \in \Delta(\sigma)} \frac{w(\delta)}{w(\sigma)} P_L(\delta(\sigma), \tau) \tag{2.4}$$

Remarque 2.8. *Le fait qu'aucune configuration du PLCS ne soit bloquante garantit que* $\mathbf{P}(\cdot,\cdot)$ *est bien définie (pas de division par zéro dans l'équation* (2.4)*).*

On vérifie que la définition de $\mathbf{P}$ fait bien de $\mathcal{M}_\mathcal{P}$ une chaîne de Markov. Pour cela, montrons que $\sum_\tau \mathbf{P}(\sigma,\tau) = 1$.

$$\begin{aligned}
\sum_\tau \mathbf{P}(\sigma,\tau) &= \sum_\tau \sum_{\delta \in \Delta(\sigma)} \frac{w(\delta)}{w(\sigma)} P_L(\delta(\sigma),\tau) \\
&= \frac{1}{w(\sigma)} \sum_{\delta \in \Delta(\sigma)} w(\delta) \sum_\tau P_L(\delta(\sigma),\tau) \\
&= \frac{1}{w(\sigma)} \sum_{\delta \in \Delta(\sigma)} w(\delta) && \text{grâce à (2.3)} \\
&= 1 && \text{par définition de } w(\sigma)
\end{aligned}$$

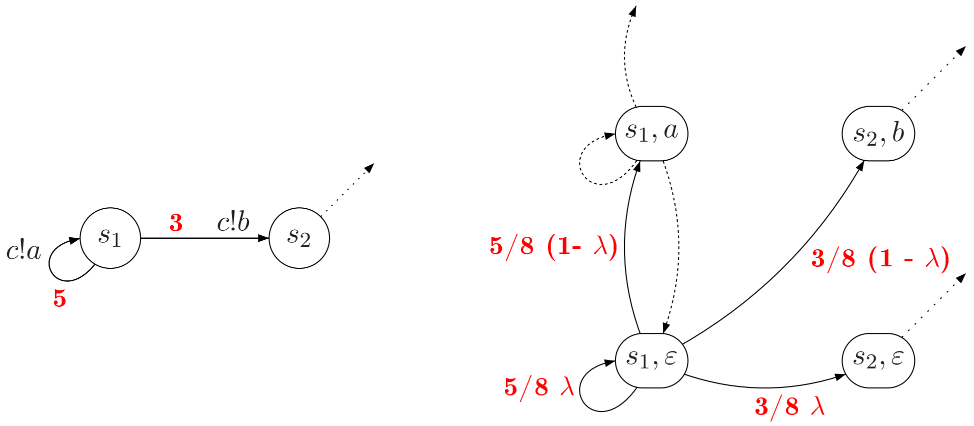

FIG. 2.4 – La sémantique markovienne sur un exemple

Exemple 2.9. *La Figure 2.4 illustre la définition de $\mathcal{M}_\mathcal{P}$ à partir de $\mathcal{P}$. On a isolé une partie du PLCS $\mathcal{P}$, constituée de deux états de contrôle s_1 et s_2. Deux transitions partent de l'état de contrôle s_1. On représente à droite de la figure la partie de la chaîne de Markov correspondant à l'application d'une transition à partir de la configuration (s_1, ε).*

2.2 Notion d'attracteur dans les chaînes de Markov

2.2.1 Présentation

Dans cette sous-section, nous introduisons la notion d'*attracteur* pour les chaînes de Markov. Nous verrons plus tard que les attracteurs (et plus précisément les attracteurs finis) jouent un rôle important dans l'analyse des chaînes de Markov. L'existence d'un attracteur fini dans une chaîne de Markov infinie simplifie en effet l'étude des propriétés de cette chaîne et rapproche son analyse de celle d'une chaîne de Markov finie. Nous appliquerons les techniques exposées pour les chaînes de Markov possédant un attracteur fini à la vérification qualitative de propriétés d'accessibilité ou d'accessibilité répétée pour les PLCS.

Définition 2.10 (Attracteur)**.** *Un ensemble $A \subseteq E$ d'états de la chaîne de Markov $\mathcal{M} = (E, \mathbf{P})$ est un* attracteur *si pour tout état $x \in E$ on a $\Pr(x \models \Diamond A) = 1$.*

Autrement dit, un attracteur est en ensemble d'état qui sera visité presque sûrement, quelque soit l'état initial. Toute chaîne de Markov admet pour attracteur l'ensemble E de tous ses états ; ce dernier sera appelé *attracteur trivial*.

Proposition 2.11. *Soit A un attracteur pour $\mathcal{M} = (E, \mathbf{P})$. Alors pour tout état $x \in E$, on a* $\Pr(x \models \Box\Diamond A) = 1$.

Idée de la preuve. Par définition de l'attracteur, partant de x, A sera presque sûrement atteint. À chaque fois qu'une exécution quitte A, par exemple dans un état $y \in E$, le même argument permet d'assurer qu'on reviendra à A avec probabilité 1. Alors, à partir de x on visitera l'ensemble A au moins à deux reprises presque sûrement. En itérant ce raisonnement, presque sûrement l'ensemble A sera visité infiniment souvent. $\square$

Exemple 2.12 (Marches aléatoires sur $\mathbb{Z}^n$)**.** *Les marches aléatoires sur $\mathbb{Z}^n$ sont des cas particuliers de chaînes de Markov. Nous donnons ici des résultats classiques que l'on peut retrouver* *par exemple dans [Bré99]. La marche aléatoire isotrope sur $\mathbb{Z}$ représentée Figure 2.5 possède l'état $\{0\}$ comme attracteur. Presque sûrement toutes les exécutions dans cette chaîne de Markov reviennent à l'état initial* 0*. En fait, tout ensemble fini est un attracteur pour la marche aléatoire isotrope sur $\mathbb{Z}$. C'est aussi le cas pour la marche aléatoire sur $\mathbb{Z}^2$ (cf. Figure 2.6). Pour des dimensions supérieures, ceci n'est plus vrai.*

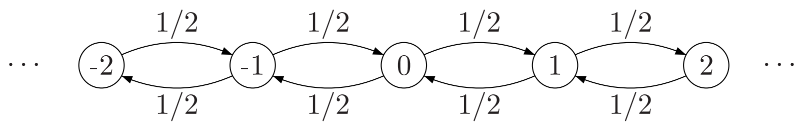

FIG. 2.5 – Marche aléatoire isotrope sur $\mathbb{Z}$

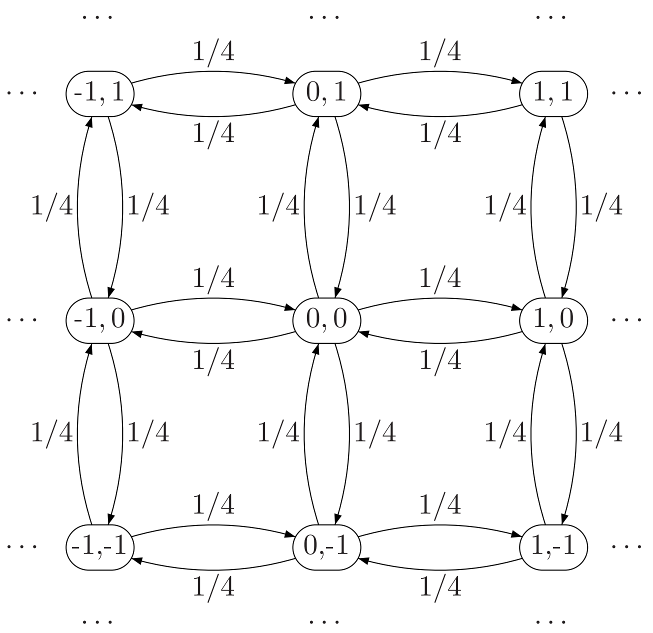

FIG. 2.6 – Marche aléatoire isotrope sur $\mathbb{Z}^2$

Lemme 2.13. *Soit A un attracteur d'une chaîne de Markov $\mathcal{M}$, et A' un ensemble d'états de $\mathcal{M}$. On suppose que A est* fini*. Si l'ensemble A' est accessible depuis chaque état $x \in A$, alors A' est aussi un attracteur pour $\mathcal{M}$.*

Preuve : Fixons $x \in E$. Comme A est un attracteur, $\Pr(x \models \Box\Diamond A) = 1$, c'est à dire $\Pr(x \models \Box\Diamond \bigcup_{y \in A} y) = 1$. La finitude de A entraîne alors l'existence d'un état $y_1 \in A$ tel que $\Pr(x \models \Box\Diamond y_1) = 1$. (En général, cet élément y_1 dépend de l'état initial x). Par hypothèse, y_1 peut atteindre A', donc il existe $y_2 \in A'$ tel que $y_1 \xrightarrow{*} y_2$. Le Lemme 2.4 (page 42) implique alors $\Pr(x \models \Box\Diamond y_2) = 1$. Comme $y_2 \in A'$, on conclut $\Pr(x \models \Box\Diamond A') = 1$. Ceci étant valable pour tout état $x \in E$, A' est un attracteur pour $\mathcal{M}$. $\square$

Dans le Chapitre 3 nous donnerons une condition suffisante pour qu'un ensemble A soit un attracteur. On s'intéresse maintenant aux conséquences de l'existence d'un attracteur fini dans les chaînes de Markov.

Les LCS probabilistes fournissent un exemple de chaînes de Markov qui possèdent un attracteur fini :

Théorème 2.14. *Soit $\mathcal{P}$ un PLCS, et $S_0 \subseteq \mathsf{Conf}$ l'ensemble des configurations pour lesquelles les canaux sont vides. Alors S_0 est un attracteur dans $\mathcal{M}_P$.*

La preuve de ce théorème sera donnée dans le Chapitre 3 comme conséquence du Théorème 3.5 (page 62). Néanmoins, nous allons utiliser ce fait dans la suite, c'est pourquoi nous l'énonçons dès maintenant.

Intuitivement lorsqu'un nombre important de messages est en transit dans les canaux, la probabilité pour qu'à la prochaine étape le nombre de messages diminue est plus forte que la probabilité que le nombre de messages augmente. Ainsi le système sera ramené à des configurations petites (ayant peu de configurations), et en particulier à celles où les canaux sont vides.

Le concept d'attracteur pour les PLCS est introduit indépendamment dans [AR03] et [BS03]. La notion est sous-jacente dans [BE99], sans être explicitée. Néanmoins, c'est la clef de la vérification qualitative des PLCS. Depuis, la notion d'attracteur dans les chaînes de Markov est beaucoup utilisée, voire raffinée [ABM05, ABMS06].

2.3 Vérification qualitative des PLCS

2.3.1 Vérification qualitative dans les chaînes de Markov : accessibilité et accessibilité répétée

Soit $\mathcal{M} = (E, \mathbf{P})$ une chaîne de Markov ayant $A \subseteq E$ comme attracteur fini. On définit *Graph*(A) comme le graphe dont les sommets sont les configurations du sous-ensemble $A \subseteq E$ et tel qu'il existe une arête entre deux configurations σ et τ de A si et seulement si $\sigma \xrightarrow{*} \tau$ dans $\mathsf{ST}_{\mathcal{M}}$, c'est à dire si la probabilité d'atteindre τ à partir de σ dans $\mathcal{M}$ (en une ou plusieurs étapes) est strictement positive. Cette définition fait de *Graph*(A) un sous-graphe de la fermeture transitive de $\mathsf{ST}_{\mathcal{M}}$.

Exemple 2.15. *Pour les PLCS, un attracteur fini est l'ensemble des configurations avec canaux vides S_0. On donne un exemple de la construction de Graph(S_0) pour un PLCS particulier à la Figure 2.7. À gauche figure un système communicant à un canal (non représenté) que l'on peut voir comme un PLCS en ajoutant un taux de pertes λ et des poids aux règles de transition. À droite est représenté Graph(S_0) pour ce PLCS. Notons qu'il n'y a pas d'arête entre (s_2, ε) et (s_3, ε) car depuis s_2 une lecture est nécessaire pour atteindre s_3.*

Définition 2.16. *On appelle* composante fortement connexe *d'un graphe orienté $G = (V, \rightarrow)$ un ensemble maximal de sommets $C \subseteq V$ tels que pour tout couple $(x, y) \in C$, $x \xrightarrow{*} y$ et $y \xrightarrow{*} x$.*

C est appelée composante fortement connexe finale *si C est une composante fortement connexe et pour tout $x \in C$ il n'existe pas d'arête $x \rightarrow y$ avec $y \notin C$.*

Exemple 2.17. *Les composantes fortement connexes finales de Graph(S_0) sont entourées dans la Figure 2.7 : $C_1 = \{(s_1, \varepsilon)\}$, $C_2 = \{(s_3, \varepsilon), (s_4, \varepsilon)\}$ et $C_3 = \{(s_5, \varepsilon)\}$.*

Dans un premier temps, nous donnons une propriété des composantes fortement connexes finales de *Graph*(A) lorsque A est un attracteur fini.

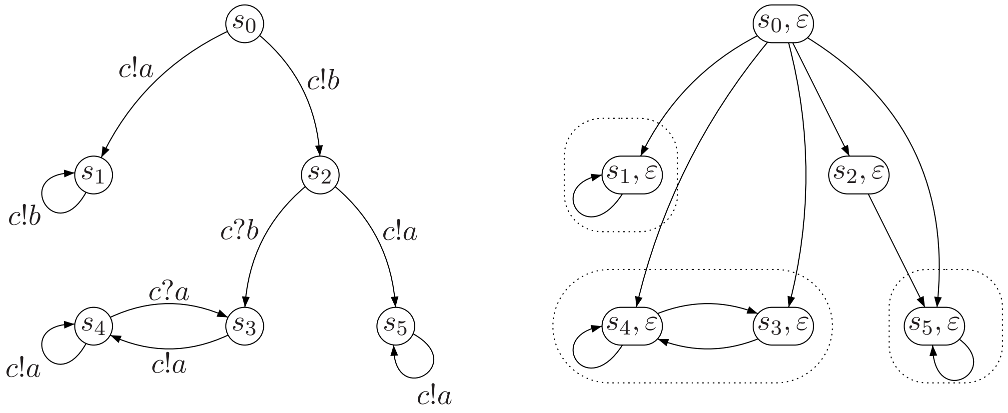

FIG. 2.7 – Exemple de PLCS et *Graph*(S_0) associé

Lemme 2.18. *Soit A un attracteur fini d'une chaîne de Markov $\mathcal{M} = (E, \mathbf{P})$. Si C est une composante fortement connexe finale de Graph(A) et x un état de C, alors pour tout état $s' \in S$,* $\Pr(x \models \Box\Diamond x') = 1$.

Preuve : A est un attracteur pour $\mathcal{M}$ donc $\Pr(x \models \Box\Diamond A) = 1$. D'autre part, puisque C est une composante fortement connexe finale de *Graph*(A), les états de $A \setminus C$ ne sont pas atteignables depuis C. Ainsi, $\Pr(x \models \Box\Diamond A) = 1$ implique $\Pr(x \models \Box\Diamond C) = 1$, car $x \in C$.

A est fini, donc C est fini également. On en déduit qu'il existe un état x' de la composante C tel que $\Pr(x \models \Box\Diamond x') = 1$. Comme tout état de C est accessible depuis tout autre état (par exemple x'), on obtient $\Pr(x \models \Box\Diamond x') = 1$ pour tout $x' \in E$, en application du Lemme 2.4, page 42. $\square$

Lemme 2.19. *Soit A un attracteur fini d'une chaîne de Markov $\mathcal{M} = (E, \mathbf{P})$. Soient $C_1, \ldots, C_p$ les composantes fortement connexes finales de Graph(A). Pour tout état $x \in E$*

$$\Pr(x \models \Diamond C_1) + \ldots + \Pr(x \models \Diamond C_p) = \Pr(x \models \Box\Diamond C_1) + \ldots + \Pr(x \models \Box\Diamond C_p) = 1.$$

Preuve : Par définition des composantes fortement connexes finales, $C \stackrel{\text{def}}{=} C_1 \cup \ldots C_p$ est atteignable depuis tout état de A. Le lemme 2.13 (cf. page 46) implique que C est un attracteur. De plus $\Pr(x \models \Diamond C_i \wedge \Diamond C_j) = 0$ si $i \neq j$ car dans ce cas C_i n'est pas accessible depuis C_j. Ainsi $\Pr(x \models \Diamond C) = 1$ entraîne $\Pr(x \models \Diamond C_1) + \ldots + \Pr(x \models \Diamond C_p) = 1$. On conclut en observant que $\Pr(x \models \Diamond C_i) = \Pr(x \models \Box\Diamond C_i)$ grâce au lemme 2.18. $\square$

Ce dernier lemme peut être raffiné en

Lemme 2.20. *Soient A un attracteur fini d'une chaîne de Markov $\mathcal{M} = (E, \mathbf{P})$, et $x \in E$ un état de $\mathcal{M}$. Si $C'_1, \ldots, C'_q$ sont les composantes fortement connexes finales de Graph(A) qui sont accessibles depuis x, alors*

$$\Pr(x \models \Diamond C'_1) + \ldots + \Pr(x \models \Diamond C'_q) = 1.$$

Preuve : C'est une conséquence du lemme 2.19, en constatant que si une composante fortement connexe finale C_i n'est pas accessible depuis x, $\Pr(x \models \Diamond C_i) = \Pr(x \models \Box\Diamond C_i) = 0$. $\square$

Enfin nous donnons un lemme qui sera utilisé pour résoudre le problème de l'accessibilité répétée probabiliste.

Lemme 2.21. *Soit A un attracteur fini de la chaîne de Markov $\mathcal{M}$. Soient $x \in E$ un état et $Q \subseteq E$ un ensemble d'états. Alors $\Pr(x \models \Box\Diamond Q) = 1$ si et seulement si depuis toute composante fortement connexe finale accessible depuis x, on peut atteindre Q.*

Remarque 2.22. *Si depuis toute composante fortement connexe finale on peut atteindre Q, alors depuis chaque état de chaque composante fortement connexe finale on peut atteindre Q.*

Preuve : ($\Longleftarrow$) : Notons $C'_1, \ldots, C'_q$ les composantes fortement connexes finales que l'on peut atteindre depuis x, et UC l'union de ces ensembles : $UC = C'_1 \cup \ldots C'_q$. Le lemme 2.20 donne $\Pr(x \models \Diamond UC) = \Pr(x \models \Box\Diamond UC) = 1$. On conclut par le lemme 2.4 (page 42) que :

$$\Pr(x \models \Box\Diamond UC \wedge \Box\Diamond Q) = 1$$

car Q est accessible depuis chaque état UC.

($\Longrightarrow$) : Supposons que Q n'est pas accessible depuis une composante fortement connexe finale donnée C. Alors $\Pr(x \models \Box\Diamond Q) \leq 1 - \Pr(x \models \Diamond C) < 1$. $\Box$

Accessibilité probabiliste qualitative Nous expliquons à présent comment vérifier des questions d'accessibilité dans les chaînes de Markov avec attracteur. Comme on va le voir, les propriétés qualitatives d'accessibilité probabiliste se réduisent à des questions d'accessibilité (non déterministe) sur le système de transition sous-jacent à la chaîne de Markov. C'est-à-dire que les valeurs précises des probabilités n'ont pas d'influence. Le fait qu'on puisse réduire le problème de l'accessibilité probabiliste à des questions d'accessibilité non déterministe est similaire à la situation des chaînes de Markov finies. Pour les chaînes de Markov infinies ce n'est pas le cas en général. Ceci n'est vrai que grâce à l'existence d'un attracteur fini, existence qui dépend des valeurs des probabilités.

Formellement, le problème que nous considérons est le suivant :

> **Accessibilité probabiliste**
> **Instance :** Une chaîne de Markov $\mathcal{M} = (E, \mathbf{P})$, un état $x \in E$, et un ensemble $Q \subseteq E$.
> **Question :** Est-ce que $\Pr(x \models \Diamond Q) = 1$?, *i.e.* est-ce que Q est atteint presque sûrement depuis x ?

Soit $\mathcal{M} = (E, \mathbf{P})$ une chaîne de Markov ayant pour attracteur fini un ensemble $A \subseteq E$. On réduit les problèmes d'accessibilité probabiliste dans $\mathcal{M}$ à des problèmes d'accessibilité dans *Graph*(A).

Lemme 2.23. *Soient $\mathcal{M}$ une chaîne de Markov, A un attracteur fini de $\mathcal{M}$, $x \in E$ un état de $\mathcal{M}$ et $Q \subseteq E$ un ensemble d'états. Alors $\Pr(x \models \Diamond Q) < 1$ si et seulement si il existe une composante fortement connexe finale C de Graph(A) telle que :*

1. *C ne peut atteindre Q, et,*
2. *on peut atteindre C depuis x sans passer par Q.*

Preuve : ($\Longleftarrow$) : Soit π un chemin fini menant de x à C sans traverser Q. Puisque par hypothèse Q n'est pas atteignable à partir de C, toute exécution ayant comme préfixe π ne rencontre jamais Q. L'ensemble de ces exécutions a pour mesure la mesure du cylindre engendré par π, c'est-à-dire $\Pr(D_\pi) > 0$. Alors $\Pr(x \models \neg\Diamond Q) \geq \Pr(D_\pi) > 0$, et donc $\Pr(x \models \Diamond Q) < 1$.

($\Longrightarrow$) : Notons $C_1, \ldots, C_p$ les composantes fortement connexes finales de $\mathit{Graph}(A)$, et UC l'union de ces ensembles : $UC = C_1 \cup \ldots C_p$. On a vu que UC est un attracteur s'il existe un attracteur fini pour $\mathcal{M}$ (cf. Lemme 2.19 page 48). On en déduit que $\Pr(x \models \Box\Diamond UC) = 1$ pour tout état $x \in E$, et donc $\Pr(x \models \Diamond Q) = \Pr(x \models \Diamond Q \wedge \Box\Diamond UC)$. Les événements $\Box\Diamond C_1, \ldots, \Box\Diamond C_p$ forment une partition de $\Box\Diamond UC$ car C_i n'est pas accessible depuis C_j quand $i \neq j$ (voir toujours le Lemme 2.19). En conséquence, $\Pr(s \models \Diamond Q \wedge \Box\Diamond UC) = \Pr(s \models \Diamond Q \wedge \Box\Diamond C_1) + \ldots + \Pr(s \models \Diamond Q \wedge \Box\Diamond C_p)$. Alors $\Pr(s \models \Diamond Q) < 1$ entraîne l'existence de $C \in \{C_1, \ldots, C_p\}$ tel que

$$\Pr(s \models \Diamond Q \wedge \Box\Diamond C) < \Pr(s \models \Box\Diamond C). \tag{2.5}$$

Raisonnons à présent par l'absurde pour montrer que C satisfait les assertions (1) et (2) du lemme. Si $C \xrightarrow{*} Q$, alors $\Pr(x \models \Box\Diamond C) = \Pr(x \models \Box\Diamond C \wedge \Box\Diamond Q)$ (cf. Lemme 2.4 page 42), qui contredit la conclusion (2.5). De façon similaire, si toutes les exécutions partant de x qui atteignent C visitaient Q avant C, alors $\Pr(x \models \Box\Diamond C) = \Pr(x \models \Box\Diamond C \wedge \Diamond Q)$, ce qui contredit également (2.5). Finalement l'équation (2.5) implique (1) et (2). $\Box$

Le lemme 2.23 nous donne une procédure pour résoudre le problème d'accessibilité qualitative probabiliste.

Procédure – Accessibilité probabiliste

Entrée Une chaîne de Markov $\mathcal{M} = (E, \mathbf{P})$ ayant un attracteur fini, et son système de transition sous-jacent $\mathsf{ST}_\mathcal{M} = (E, \rightarrow)$, un état $x \in E$, et un ensemble $Q \subseteq E$.
Sortie Q est-il atteint presque sûrement depuis x ?
début
1. construire un attracteur fini A
2. construire $\mathit{Graph}(A)$ et établir la liste de ses composantes fortement connexes finales $C_1, \ldots, C_p$
3. **pour** chaque composante fortement connexe finale C de $\mathit{Graph}(A)$
   3a. **si** $x \in (S \setminus Q)$ *Until* C
   3b. **et** $\neg(C \xrightarrow{*} Q)$
   3c. **alors** retourner(faux)
4. retourner(vrai)

fin

Accessibilité répétée probabiliste Le problème de l'accessibilité répétée probabiliste [2] est défini ci-dessous.

[2] Ici encore, on ne considère que la variante « qualitative »

Accessibilité répétée probabiliste
Instance : Une chaîne de Markov $\mathcal{M} = (E, \mathbf{P})$ ayant un attracteur fini, et son système de transition sous-jacent $\mathsf{ST}_{\mathcal{M}}$, un état $x \in E$, et un ensemble $Q \subseteq E$.
Question : Est-ce que $\Pr(x \models \Box\Diamond Q) = 1$?, *i.e.* est-ce que Q est visité infiniment souvent presque sûrement depuis x ?

Le lemme 2.21 permet de définir une procédure résolvant le problème de l'accessibilité répétée probabiliste.

Procédure – Accessibilité répétée probabiliste

Entrée Une chaîne de Markov $\mathcal{M} = (E, \mathbf{P})$ ayant pour système de transition sous-jacent $T = (E, \to)$, un état $x \in E$, et un ensemble $Q \subseteq E$.
Sortie Q est-il visité infiniment souvent presque sûrement depuis x ?
début
1. construire un attracteur fini A
2. construire *Graph*(A) et établir la liste de ses composantes fortement connexes finales $C_1, \ldots, C_p$
3. **pour** chaque composante fortement connexe finale C de *Graph*(A)
   3a. **si** $x \xrightarrow{*} C$
   3b. **et** $\neg\big(C \xrightarrow{*} Q\big)$
   3c. **alors** retourner(faux)
4. retourner(vrai)

fin

Cette méthode est calquée sur celle proposée pour l'accessibilité probabiliste et ne diffère que par l'instruction 3a. qui est remplacée par :

3a. **si** $x \xrightarrow{*} C$

2.3.2 Application aux PLCS

Accessibilité et accessibilité répétée Puisque l'accessibilité est décidable dans les LCS et que le système de transition sous-jacent à un PLCS est indépendant du taux de perte λ, on obtient :

Lemme 2.24. *Étant donné un ensemble fini de configurations A, on peut construire de façon effective Graph(A).*

De plus, si on se donne deux PLCS $\mathcal{P} = \langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w\rangle$ et $\mathcal{P}' = \langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda', w'\rangle$ qui ne diffèrent que par les valeurs des probabilités (c'est-à-dire on suppose que pour tout $\delta \in \Delta,\ w(\delta) > 0 \iff w'(\delta) > 0$), *Graph*$(A)$ sera identique pour $\mathcal{P}$ et $\mathcal{P}'$. Alors, dans les problèmes d'accessibilité probabiliste et accessibilité répétée probabiliste, qui sont basées sur une analyse de *Graph*(A), les réponses seront les mêmes pour $\mathcal{P}$ et $\mathcal{P}'$, sous réserve que A soit un attracteur dans les deux cas. Soulignons le fait que ceci est lié au fait que $\mathcal{P}$ et $\mathcal{P}'$ possèdent un même attracteur fini. On retrouve ici un fait classique du cas où la chaîne de Markov est finie : lorsque l'on s'intéresse à des propriétés *qualitatives*, la valeur exacte des probabilités n'a pas d'importance, seule la topologie du graphe compte. C'est pourquoi dans l'exemple 2.15 (page 47) il n'était pas utile de préciser des valeurs pour le taux de pertes λ et les poids des différentes règles de transition.

Définissons formellement les problèmes d'accessibilité et accessibilité répétée pour les PLCS.

Accessibilité probabiliste dans les PLCS
Instance : Un PLCS $\mathcal{P} = \langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w \rangle$, une configuration $\sigma \in \mathsf{Conf}$, et une région régulière $R \subseteq \mathsf{Conf}$.
Question : Est-ce que $\Pr(\sigma \models \Diamond R) = 1$?, *i.e.* est-ce que la région R est accessible presque sûrement depuis σ ?

Accessibilité répétée probabiliste dans les PLCS
Instance : Un PLCS $\mathcal{P} = \langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w \rangle$, une configuration $\sigma \in \mathsf{Conf}$, et une région régulière $R \subseteq \mathsf{Conf}$.
Question : Est-ce que $\Pr(\sigma \models \Box\Diamond R) = 1$?, *i.e.* est-ce que la région R est visitée infiniment souvent presque sûrement depuis σ ?

Théorème 2.25. *Les problèmes de l'accessibilité probabiliste et de l'accessibilité répétée probabiliste sont décidables pour les PLCS.*

Preuve : Les Lemmes 3.6 (page 64) et 2.24 (page 51) et la décidabilité du problème de l'accessibilité dans les LCS (cf. Corollaire 1.30 page 34) permettent d'implémenter la procédure d'accessibilité répétée probabiliste dans les chaînes de Markov. Pour ce qui est de l'accessibilité probabiliste, on peut utiliser le fait que les problèmes d'accessibilité contrainte sont décidables dans les LCS (c'est aussi une conséquence du Théorème 1.26 sur les termes du μ-calcul gardé).

Il existe une autre preuve : réduire l'accessibilité probabiliste à l'accessibilité répétée probabiliste. Nous détaillons cette réduction ci-dessous. Soit $\mathcal{P}$ un PLCS, et $R\mathsf{Conf}$ une région. On modifie la structure de $\mathcal{P}$ pour obtenir un PLCS étendu $\mathcal{P}'$. Si $s \xrightarrow{op} t$ est une transition de $\mathcal{P}$, elle est remplacée dans $\mathcal{P}'$ par la transition gardée $s \xrightarrow{\overline{R}:op} t$. De plus pour tout état de contrôle s, on ajoute une règle de transition $s \xrightarrow{R:op} s$. Cette construction vérifie l'équivalence suivante :

$$\Pr(\sigma \models_{\mathcal{P}'} \Box\Diamond R) = \Pr(\sigma \models_{\mathcal{P}} \Diamond R) \tag{2.6}$$

ce qui montre que le problème de l'accessibilité se réduit au problème d'accessibilité répétée dans les PLCS. $\square$

Exemple 2.26. *Reprenons l'exemple de la Figure 2.7 page 48 (et les notations de l'Exemple 2.17), avec* $R = (s_1, \uparrow b) + (s_4, \uparrow\varepsilon) + (s_5, \uparrow aa)$*, et* $x = (s_0, \varepsilon)$ *comme état initial. Alors depuis chacune des composantes fortement connexes finales de* $GraphS_0$*, on peut atteindre R. Ainsi,* $\Pr((s_0, \varepsilon) \models \Diamond R) = 1$ *et aussi* $\Pr((s_0, \varepsilon) \models \Box\Diamond R) = 1$

On considère à présent la région $R' = (s_1, \uparrow b) + (s_2, \uparrow\varepsilon)$*. Depuis* C_1 *on peut atteindre* R' *ce qui viole la condition* 3.*b (voir la procédure d'accessibilité probabiliste page 50) ; pour* C_2 *et* C_3*, c'est la condition* 3.*a qui n'est pas respectée puisque* s_2 *est visité avant d'atteindre* C_2 *ou* C_3*. On conclut que* $\Pr((s_0, \varepsilon) \models \Diamond R') = 1$*. Par contre, si on s'intéresse à la propriété* $\Box\Diamond R'$*,* C_2 *et* C_3 *sont accessibles depuis* (s_0, ε) *et satisfont* $\neg C_i \xrightarrow{*} Q$*. Autrement dit, à la fois* C_2 *et* C_3 *satisfont les conditions* 3.*a et* 3.*b de la procédure pour l'accessibilité répétée probabiliste proposée page 51. On en déduit que* $\Pr((s_0, \varepsilon) \models \Box\Diamond R') < 1$.

Model-checking probabiliste On s'intéresse maintenant à des propriétés plus générales que l'accessibilité ou l'accessibilité répétée pour les PLCS. Soit ϕ une formule du temps linéaire et $\mathcal{P}$ un PLCS. On se demande si $\Pr(\sigma \models \phi) = 1$ pour une configuration σ donnée, c'est-à-dire si les exécutions issues de σ satisfont presque sûrement la propriété ϕ.

Nous montrons ici que si les propriétés sont spécifiées par des automates fini (avec conditions d'acceptation de Müller par exemple), ou de façon équivalente par des formules de la logique monadique du second ordre (formules MSO), le problème est décidable.

Des résultats similaires sont obtenus pour des PLCS ayant d'autres types d'erreurs en plus des pertes (duplication, corruption ou insertion de messages).

Remarque 2.27. *Nous supposons ici que les propositions atomiques des formules du temps linéaire sont des régions de contrôle, plutôt que des régions en toute généralité. Cette restriction permet de donner une preuve plus simple mais le résultat de décidabilité tiendrait pour des formules LTL sur des régions régulières. Pour cela, il serait néanmoins nécessaire d'utiliser des LCS étendus (cf. section 1.2 page 26), pour réaliser le produit d'un automate de Müller pour la propriété avec le LCS sous-jacent au NPLCS considéré. Nous traitons donc dans la suite le cas plus simple où les proposition atomiques des formules du temps linéaires sont des régions de contrôle (*i.e. *des ensembles d'états de contrôle).*

Pour vérifier une propriété spécifiée par un automate fini déterministe, il nous faut construire le produit de cet automate avec le PLCS. Cette approche nécessite d'étendre les LCS avec une fonction d'étiquetage. Un *LCS étiqueté* est un LCS muni d'un alphabet Σ et d'une fonction d'étiquetage $l : S \to \Sigma$. L'étiquetage d'un LCS $\mathcal{L}$ est transféré au système de transitions $\mathsf{ST}_\mathcal{L}$ sous-jacent à $\mathcal{L}$. Une configuration $\sigma = (s, \mathbf{w})$ a pour étiquette celle de son état de contrôle : $l(\sigma) = l(s)$. Pour un PLCS $\mathcal{P} = \langle \mathcal{L}, \lambda, w \rangle$, on peut également supposer que le LCS sous-jacent $\mathcal{L}$ est muni d'un étiquetage et que ce dernier est étendu à la chaîne de Markov $\mathcal{M}_\mathcal{P}$. De cette manière, on obtient des PLCS étiquetés, et des chaînes de Markov étiquetées.

La trace d'une exécution $\sigma_0\sigma_1\cdots$ dans un système de transition étiqueté est le mot $l(\sigma_0)l(\sigma_1)\cdots \in \Sigma^\omega$. Une exécution vérifie la propriété spécifiée par l'automate $\mathcal{A}$, si sa trace est acceptée par $\mathcal{A}$.

Définition 2.28 (Automate de Müller)**.** *Un* automate de Müller *est un n-uplet* $\mathcal{A} = \langle Q, \Sigma, \to, q_0, \mathcal{F} \rangle$ *où*

- Q *est un ensemble fini d'états,*
- Σ *est un alphabet fini,*
- $\to \subseteq Q \times \Sigma \times Q$ *est la relation de transition,*
- $q_0 \in Q$ *est l'état initial, et*
- $\mathcal{F} \subseteq 2^Q$ *est un ensemble de conditions d'équité.*

Classiquement, on écrit $q \xrightarrow{a} q'$ si $(q, a, q') \in \to$. $\mathcal{A}$ est *déterministe* si pour tout état $q \in Q$ et toute lettre $a \in \Sigma$, il y a exactement un état $q' \in Q$ tel que $q \xrightarrow{a} q'$.

Une *exécution* de $\mathcal{A}$ est une suite infinie $q_0a_0q_1a_1\cdots$ telle que pour tout $i \geq 0$, $q_i \xrightarrow{a_i} q_{i+1}$. À une exécution π de $\mathcal{A}$, on associe l'ensemble $\mathtt{Inf}(\pi)$ des états $q \in Q$ visités infiniment souvent le long de π. Une exécution π satisfait les conditions d'équité $\mathcal{F}$ si l'ensemble $\mathtt{Inf}(\pi)$ appartient à $\mathcal{F}$. Un mot infini $a_0a_1\cdots$ sur Σ est accepté par $\mathcal{A}$ s'il existe une exécution $q_0a_0q_1a_1\cdots$ qui satisfait les conditions d'équité $\mathcal{F}$. Le *langage accepté par* $\mathcal{A}$ est l'ensemble des mots infinis acceptés par $\mathcal{A}$.

On rappelle que pour tout langage défini par une formule de MSO, il existe un automate de Müller déterministe dont c'est le langage accepté [Tho90]. Ceci permet de se restreindre aux automates de Müller déterministes pour spécifier les propriétés.

Soit $\mathcal{A} = \langle Q, \Sigma, \rightarrow, q_0, \mathcal{F}\rangle$ un automate de Müller et $\mathsf{ST} = (S, \rightarrow, \Sigma, l)$ un système de transition étiqueté. Le produit $\mathcal{A} \times \mathsf{ST}$ de $\mathcal{A}$ et ST est un système de transition $\mathsf{ST}' = \langle S', \rightarrow', q_0', l'\rangle$ défini de la façon suivante :

États : $S' = Q \times S$,

Étiquetage : $l'(q, s) = l(s)$; (q, s) a la même étiquette que s dans ST, et

Relation de transition : $(q, s) \rightarrow' (q', s')$ ssi $s \rightarrow s'$ dans ST et $q \xrightarrow{l(s)} q'$ dans $\mathcal{A}$.

On définit également, le produit $R = \mathcal{A} \times \mathcal{M}$ d'un automate déterministe $\mathcal{A}$ avec une chaîne de Markov étiquetée $\mathcal{M} = \langle E, \mathbf{P}, \Sigma, l\rangle$. R est une chaîne de Markov étiquetée $\langle E', \mathbf{P}', \Sigma, l'\rangle$. Les états de R sont les couples $(q, x) \in Q \times E$, et la fonction d'étiquetage l' vérifie $l'(q, x) = l(x)$. Enfin, la probabilité $\mathbf{P}'$ est donnée par :

$$\mathbf{P}'((q, x), (q', x')) = \begin{cases} \mathbf{P}(x, x') & \text{si } q \xrightarrow{l(s)} q' \text{ dans } \mathcal{A}, \\ 0 & \text{sinon.} \end{cases}$$

Remarque 2.29. *Le fait que $\mathcal{A}$ soit déterministe assure que la somme des probabilités des transitions à partir d'un état (q, x) est la même que la somme des probabilités des transitions à partir de l'état x dans $\mathcal{M}$, et donc cette somme est 1. Ainsi, R est bien une chaîne de Markov étiquetée.*

De plus, $\mathcal{A} \times \mathsf{ST}_{\mathcal{M}}$ est le système de transition sous-jacent à $\mathcal{A} \times \mathcal{M}$: $\mathcal{A} \times \mathsf{ST}_{\mathcal{M}} = \mathsf{ST}_{\mathcal{A}\times\mathcal{M}}$.

Enfin, le produit $\mathcal{L}' = \mathcal{A} \times \mathcal{L}$ d'un automate avec un LCS $\mathcal{L} = \langle S, \mathsf{C}, \mathsf{M}, \Delta, \Sigma, l\rangle$ est défini par : $\mathcal{L}' = \langle S', \mathsf{C}, \mathsf{M}, \Delta', \Sigma, l'\rangle$ tel que
- $S' = Q \times S$,
- $((q, s), op, (q', s')) \in \Delta'$ ssi $(s, op, s') \in \Delta$ et $q \xrightarrow{l(s)} q'$ dans $\mathcal{A}$, et
- $l'((q, s)) = l(s)$.

Le produit d'un automate déterministe avec un PLCS $\mathcal{P}$ suit le même schéma.

La notation $\mathtt{Inf}(\pi)$ est étendue aux exécutions du produit d'un automate, avec une chaîne de Markov ou un système de transition. Néanmoins, on se restreindra à la projection des exécutions sur les états de $\mathcal{A}$ lorsqu'on s'intéresse aux états visités infiniment souvent : $\mathtt{Inf}(\pi) \subseteq Q$.

Ces constructions vérifient la propriété cruciale suivante.

Lemme 2.30. *Soit $\mathcal{L}$ un LCS, et $\mathcal{P}$ un PLCS. Alors*
- *$\mathcal{A} \times \mathsf{ST}_{\mathcal{L}}$ est isomorphe à $\mathsf{ST}_{\mathcal{A}\times\mathcal{L}}$,*
- *$\mathcal{A} \times \mathcal{M}_{\mathcal{P}}$ est isomorphe à $\mathcal{M}_{\mathcal{A}\times\mathcal{P}}$.*

L'isomorphisme en question associe dans le premier cas $((q, s), \mathbf{w})$ état de $\mathsf{ST}_{\mathcal{A}\times\mathcal{L}}$ à $(q, (s, \mathbf{w}))$ état de $\mathcal{A} \times \mathsf{ST}_{\mathcal{L}}$, et dans le second cas il associe $((q, s), \mathbf{w})$ état de $\mathcal{M}_{\mathcal{A}\times\mathcal{P}}$ à $(q, (s, \mathbf{w}))$ état de $\mathcal{A} \times \mathcal{M}_{\mathcal{P}}$.

Lemme 2.31. *Soit $\mathcal{A}$ un automate déterministe ayant $\mathcal{F}$ comme conditions d'équité. Soient $\mathcal{M}$ une chaîne de Markov étiquetée, et R le produit $\mathcal{A} \times \mathcal{M}$. La mesure de l'ensemble des exécutions dans $\mathcal{M}$ à partir de x et acceptées par $\mathcal{A}$ égale la mesure de l'ensemble des exécutions π de R à partir de (q_0, x) telles que $\mathtt{Inf}(\pi) \in \mathcal{F}$.*

Preuve : À une exécution $\pi = x_0 x_1 \cdots$ de $\mathcal{M}$ on associe $\pi^{\mathcal{A}}$, l'unique exécution de R de la forme $(q_0, x_0)(q_1, s_1) \cdots$ avec pour tout i, $q_i \xrightarrow{l(x_i)} q_{i+1}$. Étant donné π, l'existence et l'unicité de $\pi^{\mathcal{A}}$ sont des conséquences du déterminisme de $\mathcal{A}$. De plus, toute exécution de R est le $\pi^{\mathcal{A}}$ d'un π de $\mathcal{M}$. Finalement, la mesure d'un ensemble L d'exécutions dans $\mathcal{M}$ est exactement la mesure dans R de $L^{\mathcal{A}} \stackrel{\text{def}}{=} \{\pi^{\mathcal{A}} | \pi \in L\}$. Le lemme découle alors de l'observation suivante : π est accepté par $\mathcal{A}$ ssi $\texttt{Inf}(\pi^{\mathcal{A}}) \in \mathcal{F}$. $\square$

Pour plus de détails sur le produit d'une chaîne de Markov avec un automate fini de mots infinis, nous suggérons de consulter [CY95] (section 4.1) où ce sujet est développé.

À présent, définissons formellement le problème qui nous intéresse.

<u>**Model-checking probabiliste dans les PLCS**</u>
Instance : Un PLCS étiqueté $\mathcal{P} = \langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w, \Sigma, l\rangle$ qui définit une chaîne de Markov étiquetée $\mathcal{M}_{\mathcal{P}}$, une configuration $\sigma \in \mathsf{Conf}$, et un automate $\mathcal{A}$.
Question : Est-ce que les exécutions de $\mathcal{M}_{\mathcal{P}}$ partant de σ sont acceptées pas $\mathcal{A}$ avec probabilité 1 ?

Ce qui reste de cette section est dédié à la démonstration du théorème :

Théorème 2.32. *Le problème du model-checking probabiliste est décidable pour les PLCS.*

Preuve : Soit $\mathcal{A} = \langle Q, \Sigma, \rightarrow, q_0, \mathcal{F}\rangle$ un automate avec conditions d'acceptation de Müller, que l'on suppose déterministe (rappelons que pour les automates de Müller ceci n'est pas une restriction). On considère le produit de $\mathcal{A}$ avec la chaîne de Markov $\mathcal{M}_{\mathcal{P}}$ induite par $\mathcal{P}$; ce produit est noté R. Grâce au Lemme 2.31, il suffit de montrer que $\Pr((q_0, \sigma) \models \texttt{Inf} \in \mathcal{F}) = 1$. Pour cela, on utilise le résultat suivant :

Lemme 2.33. *Supposons que B est un attracteur fini de R. On a alors l'équivalence entre les deux assertions :*

1. $\Pr((q_0, \sigma) \models \texttt{Inf} \in \mathcal{F}) = 1$.
2. *Pour toute composante fortement connexe finale C de $Graph(B)$, si C est accessible depuis (q_0, σ), alors il existe $F \in \mathcal{F}$ tel que :*
   - *(a) si (q, τ) est accessible dans R depuis C, alors $q \in F$, et*
   - *(b) pour tout $q \in F$, il existe une configuration τ de $\mathcal{M}_{\mathcal{P}}$ telle que (q, τ) est accessible dans R depuis C.*

Preuve du lemme : $(1 \Rightarrow 2)$: Supposons que C est une composante fortement connexe finale de $Graph(B)$, accessible depuis (q_0, σ). Alors $\Pr((q_0, \sigma) \models \Diamond C) > 0$ et $\Pr((q_0, \sigma) \models \Box\Diamond C) > 0$. Par hypothèse $\Pr((q_0, \sigma) \models \texttt{Inf} \in \mathcal{F}) = 1$, donc $\Pr((q_0, \sigma) \models \texttt{Inf} \in \mathcal{F} \wedge \Box\Diamond C) > 0$, ce qui entraîne l'existence de $F \in \mathcal{F}$ tel que

$$\Pr((q_0, \sigma) \models \texttt{Inf} = F \wedge \Box\Diamond C) > 0 \tag{2.7}$$

Ceci impose que pour tout $q \in F$, C peut atteindre (q, τ) pour une certaine configuration τ de $\mathcal{M}_{\mathcal{P}}$. De plus, si $C \xrightarrow{*} (q, \tau)$ alors $\Pr((q_0, \sigma) \models \Box\Diamond C \Leftrightarrow \Box\Diamond(q, \tau)) = 1$. C'est pourquoi (2.7) implique $q \in F$.

$(2 \Rightarrow 1)$: Supposons que pour une composante fortement connexe finale C, il existe $F \in \mathcal{F}$ vérifiant 2.(a) et 2.(b). De 2.(a) on tire $\Pr(\Box\Diamond C) = \Pr(\Box\Diamond C \wedge \texttt{Inf} = F)$. D'un autre côté, pour tout $q \in F$, 2.(b) entraîne $\Pr(\Box\Diamond C) = \Pr(\Box\Diamond C \wedge \Box\Diamond(q, \tau)) = \Pr(\Box\Diamond C \wedge q \in$

$\mathtt{Inf}$). Alors $\Pr(\Box\Diamond C) = \Pr(\Box\Diamond C \wedge \mathtt{Inf} \in \mathcal{F})$. Puisque ceci est vrai pour toute composante fortement connexe finale atteignable à partir de (q_0, σ), on obtient que $\Pr(\mathtt{Inf} \in \mathcal{F}) = 1$ par le Lemme 2.19 (page 48) **fin de preuve du lemme**

$R = \mathcal{A}\times\mathcal{M}_{\mathcal{P}}$ peut aussi être vu comme la chaîne de Markov induite par le PLCS $\mathcal{P}' = \mathcal{A}\times\mathcal{P}$ (cf. Lemme 2.30, page 54). Dans cette construction l'ensemble B des configurations de R avec canaux vides est un attracteur fini.

On peut donc appliquer le lemme 2.33 et donner des conditions nécessaires et suffisantes pour que les exécutions de $\mathcal{M}_{\mathcal{P}}$ partant de σ soient acceptées par $\mathcal{A}$ avec probabilité 1.

Il reste à montrer que les conditions du lemme 2.33 peuvent être vérifiées de façon effective. Tout d'abord, $\mathit{Graph}(B)$ est calculable par analyse d'accessibilité dans $\mathcal{P}'$. Ensuite les conditions de l'assertion 2 du lemme peuvent être vérifiées par le biais d'algorithmes pour l'accessibilité des ensembles fermés vers le haut. Plus précisément, la condition (a) demande que $((\mathcal{Q}\setminus F)\times\mathsf{Conf})\uparrow$ ne soit pas accessible, et la condition (b) demande que chaque $(\{q\}\times\mathsf{Conf})\uparrow$ soit accessible. Ces questions sont décidables pour les LCS, comme conséquences du théorème sur le mu-calcul gardé (cf. Théorème 1.26, page 32 et ses applications aux LCS). $\square$

Complexité La méthode exposée pour l'accessibilité probabiliste ou l'accessibilité répétée probabiliste d'une région dans un PLCS réduit le problème à un nombre polynomial de questions d'accessibilité dans le LCS sous-jacent. De plus, on peut aisément réduire le problème de l'accessibilité dans un LCS à une question d'accessibilité probabiliste dans un PLCS associé. Ceci permet de conclure que la vérification de propriétés d'accessibilité qualitative probabiliste dans les PLCS, et celle de propriétés d'accessibilité dans les LCS ont la même complexité. Comme l'accessibilité dans les LCS est un problème non primitif récursif [Sch02], il en est de même de l'accessibilité qualitative probabiliste dans les PLCS.

Pour ce qui est de la vérification de propriétés ω-régulières, étant donné un automate de Muller $\mathcal{A}$ pour la propriété, notre méthode réduit le problème à des questions d'accessibilité dans le produit $\mathcal{A}\times\mathcal{L}$. Si l'automate spécifiant la propriété est non-déterministe, la phase de déterminisation induit un facteur exponentiel. Dans ce cas, la complexité est bornée par $f(|\mathcal{L}|\times \exp|\mathcal{A}|)$, où f est une fonction non-primitive récursive. Une question ouverte est l'existence d'autres méthodes évitant cette explosion combinatoire, c'est-à-dire ayant une complexité de la forme $f(|\mathcal{L}|)\times g(|\mathcal{A}|)$ où g est plus simple que f, par exemple élémentaire.

2.4 Travaux liés et perspectives

LCS probabilistes Purushotoman Iyer et Narasimha furent les premiers à définir un modèle de LCS probabilistes [PN97]. Dans leur modèle, les actions sont munies d'un poids, et chaque étape peut être une action du système ou la perte d'un message. Leur sémantique probabiliste est une étape intéressante vers une modélisation plus fidèle des LCS, en voyant en particulier les pertes comme des erreurs, ayant une certaine probabilité de survenir. Cependant l'analyse faite dans ce travail est fausse.

Une deuxième approche a été entreprise par Baier et Engelen [BE99] qui fournissent une analyse correcte du modèle, mais pour laquelle des restrictions sont nécessaires. En particulier, le principal résultat est la décidabilité de la vérification de formule de LTL sans opérateur Next (ou X), lorsque le taux de perte, qui représente la fréquence des pertes par rapport aux actions du système, est supérieur à $\frac{1}{2}$. Dans le cas positif, leur travail repose, sans que ce soit clairement énoncé, sur l'existence de l'attracteur fini. Pour un taux de perte inférieur à $\frac{1}{2}$, Abdulla *et* *al.* [ABPJ05] ont montré que la vérification de formules de LTL$\setminus$ X était indécidable. La preuve est en fait exposée pour un modèle légèrement différent de celui de Baier et Engelen [BE99], puisque les pertes ne se produisent qu'en tête du canal (LCS front lossy).

Perspectives Nous nous sommes attachés à l'étude de la vérification qualitative des PLCS. Cependant, il existe des travaux dans lesquels les aspects quantitatifs ont aussi été étudiés [Rab06, Sch04].

Dans le cadre de la vérification de PLCS, les questions quantitatives qui se posent sont de la forme : étant donnés ϕ une formule du temps linéaire, σ une configuration initiale, et $\epsilon > 0$ une constante d'erreur, donner à ϵ près la probabilité que ϕ soit vérifiée à partir de σ. C'est-à-dire trouver p tel que $p - \epsilon \leq \Pr(\sigma \models \phi) \leq p + \epsilon$. Beaucoup de problèmes restent ouverts dans la vérification quantitative des PLCS. Par exemple, on ne sait toujours pas si la probabilité qu'une formule soit vérifiée est un rationnel, ou même un nombre algébrique.

Nous n'allons pas nous étendre plus sur les PLCS. En effet, le modèle des chaînes de Markov possède l'avantage de modéliser de façon réaliste les pertes de messages, mais n'est pour autant le modèle le mieux adapté à la vérification de protocoles. C'est pourquoi nous introduirons dans la troisième partie un autre modèle pour les LCS sous forme de processus de décision markovien. Nous détaillerons alors les motivations de ce nouveau modèle, et en particulier son avantage par rapport aux PLCS.

Chapitre 3

Un critère suffisant d'existence d'un attracteur fini

Sommaire

Dans ce chapitre, nous donnons un critère suffisant pour l'existence d'un attracteur dans les chaînes de Markov à espace d'état dénombrable [BBS06a]. Ce critère s'applique en particulier dans le cas des chaînes de Markov engendrées par un LCS probabiliste.

3.1 Chaînes de Markov orientées à gauche

Nous introduisons tout d'abord la notion de chaîne de Markov *orientée à gauche*. On montrera par la suite que, pour ces chaînes de Markov, un attracteur existe. Intuitivement les chaînes de Markov orientées à gauche peuvent être représentées de telle sorte que dans chaque état, il est on a tendance à plus aller vers la gauche que vers la droite. C'est par exemple le cas de la marche aléatoire sur $\mathbb{N}$ décrite Figure 3.1, avec $q = 1 - p$ et $p \geq \frac{1}{2}$.

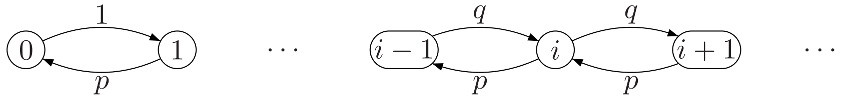

FIG. 3.1 – Exemple de marche aléatoire sur $\mathbb{N}$, orientée à gauche si $p \geq \frac{1}{2}$

Plus formellement voici la définition d'une chaîne de Markov orientée à gauche.

Soit $E = \bigcup_{i\in\mathbb{N}} S_i$ une partition disjointe de l'espace d'états de $\mathcal{M}$. L'ensemble S_i est appelé le i-ème *niveau* dans $\mathcal{M}$. Pour tout état $x \in E$, $niv(x)$, le *niveau de* x, dénote l'unique indice $i \in \mathbb{N}$ tel que $x \in S_i$.

L'espérance, partant de l'état x, du prochain niveau dans $\mathcal{M}$ est noté $\mathbb{E}(x)$. Par définition, $\mathbb{E}(x) \stackrel{\text{def}}{=} \sum_{y\in E} \mathbf{P}(x,y)niv(y)$. En regroupant les états y qui appartiennent au même niveau S_j on obtient : $\mathbb{E}(x) = \sum_{j=0}^{\infty} \mathbf{P}(x,S_j)\cdot j$.

Définition 3.1. *Étant donnée une partition $E = \bigcup_{i\in\mathbb{N}} S_i$, $\mathcal{M}$ est dite* orientée à gauche *pour $(S_i)_{i\in\mathbb{N}}$, s'il existe une constante $\delta \in \mathbb{R}^{+*}$ telle que pour tout état $x \in E\setminus S_0$, $\mathbb{E}(x) \leq niv(x)-\delta$.*

De façon informelle, une chaîne de Markov est orientée à gauche pour la partition $(S_i)_{i\in\mathbb{N}}$ si le niveau a tendance à baisser.

3.2 Résultat principal

3.2.1 Un premier critère

Théorème 3.2. *Soit $\mathcal{M}$ une chaîne de Markov orientée à gauche. Alors S_0 est un attracteur.*

Preuve : Montrons que pour tout état $x \in E$, $\Pr(x \models \Diamond S_0) = 1$. On suppose, pour faciliter les calculs, que l'ensemble S_0 forme un puits : $\forall x \in S_0,\ \mathbf{P}(x,S_0) = 1$. Cela peut se faire sans perte de généralité, car on s'intéresse aux probabilité d'atteindre S_0 ; changer les transitions sortantes de S_0 est donc sans conséquence sur le fait que S_0 est ou non un attracteur. Sous cette hypothèse, $\mathbf{P}^n(x,S_0)$ est la probabilité d'atteindre S_0 depuis x en au plus n étapes. En conséquence, $\Pr(x \models \Diamond S_0) = \lim_{n\to\infty} \mathbf{P}^n(x,S_0)$.

On commence par montrer l'inégalité suivante, concernant l'espérance du niveau après n étapes :

Lemme 3.3. *Soit δ une constante strictement positive telle que pour tout état $x \in E \setminus S_0$, $\mathbb{E}(x) \leq niv(x) - \delta$. Alors,*

$$\sum_{j=0}^{\infty} \mathbf{P}^n(x,S_j)\cdot j \;\leq\; niv(x) \;-\; n\delta \;+\; \delta\sum_{\ell=1}^{n-1} \mathbf{P}^{\ell}(x,S_0). \tag{3.1}$$

Preuve : On fait la preuve par récurrence sur n.

Pour $n = 1$, l'équation (3.1) exprime le caractère « orienté à gauche » de la chaîne de Markov $\mathcal{M}$. Soit $n \geq 2$ et supposons la propriété vraie pour $n-1$. Soit $x \in E \setminus S_0$. Alors :

$$
\begin{aligned}
\sum_{j=0}^{\infty} \mathbf{P}^n(x, S_j) \cdot j &= \sum_{j=1}^{\infty} \mathbf{P}^n(x, S_j) \cdot j && \text{(la contribution pour } j = 0 \text{ est nulle)} \\
&= \sum_{j=1}^{\infty} \sum_{k=0}^{\infty} \sum_{y \in S_k} \mathbf{P}^{n-1}(x, y) \cdot \mathbf{P}(y, S_j) \cdot j && \text{(en effet } x \xrightarrow{n} S_j \Leftrightarrow \exists y\ x \xrightarrow{n-1} y \to S_j) \\
&= \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \sum_{y \in S_k} \mathbf{P}^{n-1}(x, y) \cdot \mathbf{P}(y, S_j) \cdot j && \text{(car } S_0 \not\to S_j) \\
&= \sum_{k=1}^{\infty} \sum_{y \in S_k} \mathbf{P}^{n-1}(x, y) \cdot \sum_{j=1}^{\infty} \mathbf{P}(y, S_j) \cdot j \\
&\leq \sum_{k=1}^{\infty} \sum_{y \in S_k} \mathbf{P}^{n-1}(x, y) \cdot (niv(y) - \delta) \\
&= \sum_{k=1}^{\infty} \sum_{y \in S_k} \mathbf{P}^{n-1}(x, y) \cdot (k - \delta) \\
&= \sum_{k=1}^{\infty} \sum_{y \in S_k} \mathbf{P}^{n-1}(x, y) \cdot k \ - \ \delta \cdot \sum_{k=1}^{\infty} \sum_{y \in S_k} \mathbf{P}^{n-1}(x, y) \\
&= \sum_{k=1}^{\infty} \sum_{y \in S_k} \mathbf{P}^{n-1}(x, y) \cdot k \ - \ \delta \cdot (1 - \mathbf{P}^{n-1}(x, S_0)) \\
&\leq niv(x) \ - \ (n-1)\delta \ + \ \delta \sum_{\ell=1}^{n-2} \mathbf{P}^{\ell}(x, S_0) \ - \ \delta \cdot (1 - \mathbf{P}^{n-1}(x, S_0))
\end{aligned}
$$

par hypothèse de récurrence,

$$
= \ niv(x) \ - \ n\delta \ + \ \delta \sum_{\ell=1}^{n-1} \mathbf{P}^{\ell}(x, S_0).
$$

Ainsi $\sum_{j=0}^{\infty} \mathbf{P}^n(x, S_j) \cdot j \ \leq \ niv(x) \ - \ n\delta \ + \ \delta \sum_{\ell=1}^{n-1} \mathbf{P}^{\ell}(x, S_0)$, ce qui est exactement la propriété au rang n. □

Supposons à présent que S_0 n'est pas un attracteur pour $\mathcal{M}$. Alors il existe $x \in E$ tel que $\Pr(x \models \Diamond S_0) < 1$. Soit alors $n \in \mathbb{N}$ suffisamment grand pour que

$$
n > \frac{niv(x)}{\delta(1 - \Pr(x \models \Diamond S_0))}.
$$

L'inégalité $\mathbf{P}^\ell(s, S_0) \leq \Pr(x \models \Diamond S_0)$ entraîne

$$\begin{aligned} -n\delta + \delta \sum_{\ell=1}^{n-1} \mathbf{P}^\ell(x, S_0) &\leq -n\delta + (n-1)\delta \Pr(x \models \Diamond S_0) \\ &= -\underbrace{n\delta\big(1 - \Pr(x \models \Diamond S_0)\big)}_{> niv(x)} - \underbrace{\delta \Pr(x \models \Diamond S_0)}_{\geq 0} \\ &< -niv(x). \end{aligned}$$

Et, en appliquant l'équation (3.1), on obtient la contradiction suivante :

$$0 \leq \sum_{j=0}^{\infty} \mathbf{P}^n(x, S_j) \cdot j \leq niv(x) - n\delta + \delta \sum_{\ell=1}^{n-1} \mathbf{P}^\ell(x, S_0) < niv(x) - niv(x) = 0.$$

Cette dernière série d'inégalités mène à une absurdité. En conséquence, pour tout état $x \in E$, on a $\Pr(x \models \Diamond S_0) = 1$, et S_0 est un attracteur pour $\mathcal{M}$. □

Ce théorème permet d'unifier les preuves d'existence d'attracteur pour les PLCS et leurs variantes (cf. section 3.4, page 64). Il est apparenté au lemme de Foster (voir par exemple [Bré99], p.167). Cependant, notre théorème possède l'avantage de disposer d'une preuve élémentaire, pour laquelle aucune notion complexe de probabilité (martingales par exemple) n'est nécessaire.

L'hypothèse « orientée à gauche » pour $\mathcal{M}$ pourrait sembler trop contraignante. Nous donnons dans la sous-section 3.2.2 des extensions possibles, et dans la section 3.3 des limites du Théorème 3.2.

3.2.2 Un critère plus général

Dans un premier temps, nous relâchons l'hypothèse « orientée à gauche » pour les chaînes de Markov, en une hypothèse « faiblement orientée à gauche », qui sous réserve de faibles contraintes d'accessibilité et de cardinal, entraîne le caractère attracteur de l'ensemble S_0.

Définition 3.4. *Une chaîne de Markov $\mathcal{M}$ est dite* faiblement orientée à gauche *s'il existe une constante $\delta \in \mathbb{R}^{+*}$ et un entier $n_0 \in \mathbb{N}$ tels que pour tout état $x \in E$ dont le niveau est supérieur à n_0 (*i.e. *vérifiant $niv(x) \geq n_0$), $\mathbb{E}(x) \leq niv(s) - \delta$.*

Dans le cas des chaînes de Markov faiblement orientées à gauche, et sous quelques hypothèses supplémentaires, S_0 est encore un attracteur.

Théorème 3.5. *Soit $\mathcal{M}$ une chaîne de Markov faiblement orientée à gauche (pour un certain n_0), telle que les niveaux S_i pour $0 < i \leq n_0$ sont finis, et où S_0 est atteignable depuis tout état $x \in S_1 \cup \cdots \cup S_{n_0}$. Alors S_0 est un attracteur.*

Preuve : On considère une nouvelle partition S'_i construite à partir de S_i de la façon suivante : $S'_0 = \bigcup_{1 \leq i \leq n_0} S_i$, $S'_i = \emptyset$ pour $1 \leq i \leq n_0$ et $S'_i = S_i$ pour tout $i > n_0$. Cette construction assure que $\mathcal{M}$ est orientée à gauche pour S'_i. Le Théorème 3.2 (page 60) s'applique et S'_0 est un attracteur pour $\mathcal{M}$. Pour tout état initial $x \in E$, S'_0 est visité infiniment souvent presque sûrement. Par hypothèse, $S'_0 \setminus S_0$ est fini et S_0 est accessible depuis tout état de $S'_0 \setminus S_0$. Finalement, S_0 est visité infiniment souvent, quel que soit l'état initial. □

3.3 Limites de l'approche

Nous pouvons également poser la question du cas des chaînes de Markov orientées à droite. On pourrait penser que si la chaîne de Markov tend à aller à droite, vers des niveaux plus élevés (c'est-à-dire $\mathbb{E}(x) > niv(x)$ pour tout état x), il n'existe pas d'attracteur non trivial (*i.e.* distinct de E). Pourtant, dans ce cas, aucune conclusion générale ne peut être tirée. Et même, il n'y a aucune fonction $f : \mathbb{N} \to \mathbb{N}$ assurant qu'il n'y a pas d'attracteur non trivial si $\mathbb{E}(x) > f(niv(x))$. En effet, considérons une chaîne de Markov ayant $\mathbb{N}$ pour espace d'états, et $S_i = \{i\}$, $i = 0, 1, \ldots$ comme partition. On suppose que les probabilités de transitions satisfont $\mathbf{P}(i, 0) = \frac{1}{2}$ et $\mathbf{P}(i, 4f(i)) \geq \frac{1}{4}$. Les probabilités restantes pour les successeurs de i étant arbitraires. La Figure 3.2 présente une telle chaîne de Markov. Alors $\mathbb{E}(i) \geq f(i)$ pour tout entier i, pourtant $S_0 = \{0\}$ est un attracteur. En effet, S_0 est accessible depuis tout état $i \in S$ avec probabilité $\frac{1}{2}$.

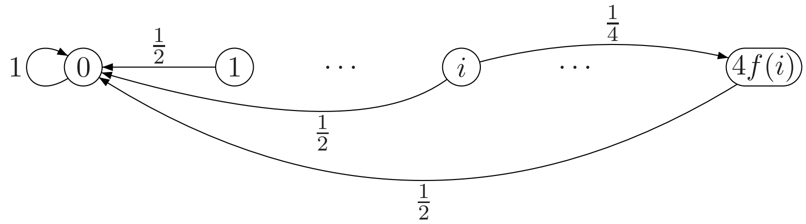

FIG. 3.2 – Exemple de chaîne de Markov avec attracteur et telle que $\mathbb{E}(x) > f(niv(x))$

D'un autre côté, il existe des chaînes de Markov telles que $\mathbb{E}(x) = niv(x) + \delta$ pour une « petite » constante δ, qui n'ont pas d'attracteur. Par exemple, considérons la marche aléatoire sur $\mathbb{N}$ représentée Figure 3.3 et formalisée par la chaîne de Markov $\mathcal{M} = (\mathbb{N}, \mathbf{P})$ avec $\mathbf{P}(i, i-1) = \frac{1}{2} - \frac{\delta}{2}$ et $\mathbf{P}(i, i+1) = \frac{1}{2} + \frac{\delta}{2}$ pour $i \geq 1$. Dans cette chaîne de Markov, $\mathbb{E}(i) = (\frac{1}{2} - \frac{\delta}{2})(i-1) + (\frac{1}{2} + \frac{\delta}{2})(i+1) = i + \delta$, mais $\mathcal{M}$ ne possède pas d'attracteur (non trivial).

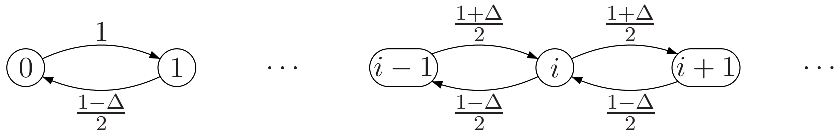

FIG. 3.3 – Marche aléatoire sur $\mathbb{N}$, dérivant à droite

Limites La condition $\delta > 0$ peut sembler trop contraignante. Nous montrons ici qu'il n'y a pas de conclusion générale dans le cas $\mathbb{E}(x) \leq niv(x)$ pour tout $x \notin S_0$.

Tout d'abord, considérons la marche aléatoire équilibrée sur $\mathbb{N}$. Plus précisément, il s'agit de la chaîne de Markov $(\mathbb{N}, \mathbf{P})$ où $\mathbf{P}(i, i+1) = \mathbf{P}(i, i-1) = \frac{1}{2}$ si $i \geq 1$ et $\mathbf{P}(0, 1) = 1$ (cf. Figure 3.4). On prend la partition induite par les états : $S_i = \{i\}$. Cette chaîne admet S_0 comme attracteur et vérifie $\mathbb{E}(i) = i$ pour tout $i > 0$.

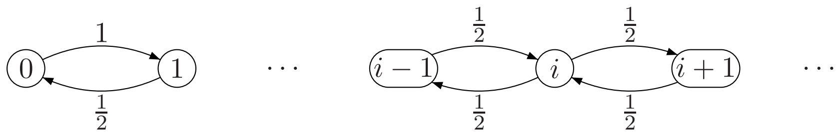

FIG. 3.4 – Marche aléatoire isotrope sur $\mathbb{N}$

Au contraire, intéressons nous à la chaîne de Markov $(\mathbb{N}, \mathbf{P})$ pour laquelle $\mathbf{P}(i, i) = 1$

décrite Figure 3.5. Pour cette chaîne de Markov, tout état vérifie $\mathbb{E}(i) = i$, mais S_0 n'est bien sûr pas un attracteur, puisqu'il n'est pas accessible depuis S_i pour $i > 0$.

FIG. 3.5 – Une marche aléatoire dégénérée sur $\mathbb{N}$

Ces deux exemples montrent l'impossibilité d'une conclusion dans le cas où $\mathbb{E}(x) \leq niv(x)$ pour tout $x \notin S_0$, sans hypothèse complémentaire.

3.4 Application aux PLCS

Nous allons montrer que dans les PLCS, l'ensemble des configurations avec canaux vides est un attracteur. C'est le résultat énoncé dans le Théorème 2.14 (page 47). L'intuition de ce résultat est la suivante : plus il y a de messages dans les canaux, plus la probabilité est forte de perdre des messages. On l'énonce à nouveau dans le lemme qui suit :

Lemme 3.6. *Soit $\mathcal{P} = \langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w\rangle$ un PLCS. Alors, $S_0 \stackrel{\text{def}}{=} \{(s, \epsilon) | s \in S\}$ est un attracteur dans $\mathcal{M}_\mathcal{P}$.*

Le résultat sur les chaînes de Markov presque orientées à gauche s'applique ici pour montrer que l'ensemble des configurations avec canaux vides est un attracteur.

Preuve : On applique le Théorème 3.5 (cf. page 62) à $\mathcal{M}_\mathcal{P}$, la chaîne de Markov induite par le PLCS $\mathcal{P}$, avec pour partition $(S_i)_{i\in\mathbb{N}}$ où S_i est l'ensemble des configurations de taille i (c'est-à-dire ayant exactement i message(s) au total dans les canaux). Montrons que cette partition fait de $\mathcal{M}_\mathcal{P}$ une chaîne presque orientée à gauche.

Soit $\sigma = (s, \mathbf{w}) \in \mathsf{Conf}$ une configuration de taille supérieure ou égale à 1 : $|\sigma| = n \geq 1$.

$$\begin{aligned}
\mathbb{E}(\sigma) &= \sum_{j=0}^{\infty} \mathbf{P}(\sigma, Q_j).j \\
&= (n + p_!(\sigma) - p_?(\sigma))(1 - \lambda) \\
&\leq (n + 1)(1 - \lambda).
\end{aligned}$$

Puisque $\lambda > 0$, il existe $N_0 \in \mathbb{N}$ tel que pour tout $n \geq N_0$

$$(n + 1)(1 - \lambda) \leq n - \frac{1}{2}.$$

Ainsi, pour tout $n \geq N_0$ et pour tout σ tel que $niv(\sigma) = n$, $\mathbb{E}(\sigma) \leq niv(\sigma) - \frac{1}{2}$. $\mathcal{M}$ est donc presque orientée à gauche et le Théorème 3.5 permet de conclure que S_0 est un attracteur. $\square$

Nous allons voir que le même résultat s'applique aux PLCS avec erreurs d'insertion et/ou de duplication.

3.5 Autres applications

3.5.1 Duplication, corruption et insertion

Dans cette section, nous nous intéressons à des PLCS pour lesquels, en plus de pertes de messages, d'autres types d'erreurs peuvent se produire, tels que la duplication, la corruption ou l'insertion de messages. Pour ces modèles qui combinent à la fois des pertes de messages et d'autres erreurs, on montre que le Théorème 3.5 s'applique encore. Ceci montre la généralité du résultat sur les chaînes de Markov orientées à gauche (ou presque orientées à gauche). Les Théorèmes 3.2 et 3.5 fournissent un critère d'existence d'attracteur fini ; jusqu'alors des preuves *ad hoc* étaient faites, lorsqu'elle étaient fournies. De plus, ces preuves spécifiques sont lourdes.

Duplication On considère une variante des PLCS où, en plus des pertes, les messages peuvent être dupliqués. Un LCS avec erreurs de duplications est encore de la forme $\mathcal{L} = \langle S, \mathsf{C}, \mathsf{M}, \Delta\rangle$ (comme un LCS). La sémantique opérationnelle est différente des LCS et prend en compte les duplications possibles.

Soit $m \in \mathsf{M}$ in message. On note m^n la concaténation de n copies du message m. Pour tout mot $x = m_1 m_2 \cdots m_n$ sur M, on définit $D(x)$ (pour « Duplication » de x) comme l'ensemble

$$\{w_1 w_2 \cdots w_n | w_i = m_i \text{ ou } w_i = m_i^2 \text{ pour tout } 1 \leq i \leq n\}$$

Ainsi $D(x)$ est donc l'ensemble des mots qui peuvent être obtenus à partir de x par duplication d'un ou plusieurs messages. On étend la définition de D à $S \times (\mathsf{C} \to \mathsf{M}^*)$ par

$$\begin{aligned} D(\mathbf{w}) : \mathsf{C} &\to \mathsf{M}^* \\ c &\to D(\mathbf{w}(c)) \end{aligned} \qquad\qquad D(s, \mathbf{w}) \stackrel{\text{def}}{=} (s, D(\mathbf{w}))$$

Remarque 3.7. *Remarquons que, si on définit l'opérateur D^{-1} par*

$$D^{-1}(\sigma) \stackrel{\text{def}}{=} \{\tau | \sigma \in D(\tau)\},$$

les extensions ensemblistes de D et D^{-1}, notés $\widehat{D} : 2^{\mathsf{Conf}} \to 2^{\mathsf{Conf}}$ et $\widehat{D^{-1}} : 2^{\mathsf{Conf}} \to 2^{\mathsf{Conf}}$, sont des opérateurs monotones, préservent la régularité et sont effectifs.

La relation de transition $\to_D$ d'un PLCS avec erreurs de duplication est définie par extension de celle $\to$ d'un PLCS classique de la façon suivante :

– Si $(s_1, \mathbf{w}_1) \xrightarrow{\delta} (s_2, \mathbf{w}_2)$ alors $(s_1, \mathbf{w}_1) \xrightarrow{\delta}_D (s_2', \mathbf{w}_2')$ pour tout $(s_2', \mathbf{w}_2') \in D((s_2, \mathbf{w}_2))$.

Il s'agit bien d'une extension, c'est-à-dire qu'il y a plus de comportements, car $\sigma \in D(\sigma)$. Cécé, Finkel et Purushothaman Iyer ont montré que l'accessibilité est décidable dans les LCS avec erreurs de duplication [CFP96].

Remarque 3.8. *La décidabilité de l'accessibilité dans les LCS avec erreur de duplication peut aussi être vue comme une conséquence du Théorème 1.26 (cf. page 32). En effet, les prédécesseurs en une étape d'un LCS avec erreurs de duplication s'expriment par un terme gardé. Soient $R \subseteq \mathsf{Conf}$ une région et $\delta \in \Delta$ une règle de transition. Alors,*

$$Pre[\delta](R) = Pre_{\text{perf}}[\delta]\big(C_\uparrow(\widehat{D^{-1}}(R))\big)$$

Comme $\widehat{D^{-1}}$ est monotone et effectif, et en utilisant la définition habituelle par point fixe de Pre^*, on peut appliquer le Théorème 1.26. Il en va de même pour l'accessibilité contrainte puisque $[\![\exists A \text{ Until } B]\!]$ s'exprime grâce à l'opérateur Pre.

Lemme 3.9. *Soit $\mathcal{L}$ un LCS avec des erreurs de duplication.*

1. *Pour toutes configurations σ_1 et σ_2, on peut décider si σ_2 est accessible à partir de s_1 [CFP96]. Le graphe $Graph(A)$ est donc calculable pour un ensemble fini de configurations donné A.*
2. *Pour toute configuration σ et tout ensemble $Q \subseteq S$ on peut décider si l'ensemble des configurations $(s, \mathbf{w})$ avec $s \in Q$ est accessible à partir de σ [CFP96].*

On probabilise maintenant un LCS avec erreurs de duplications pour donner un PLCS avec erreurs de duplications. Un PLCS avec erreurs de duplication est un n-uplet $\langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w, \lambda_D\rangle$ où $\langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w\rangle$ est un PLCS et $\lambda_D \in [0, 1]$. La valeur de λ_D représente la probabilité qu'un message soit (indépendamment des autres) dupliqué.

On définit la chaîne de Markov induite par un PLCS avec erreurs de duplication en calculant les probabilités d'atteindre des configurations par duplication de messages.

Soient $x, y \in \mathsf{M}^*$ avec $x = m_1 m_2 \cdots m_n$. On définit $\#_D(x, y)$ comme la taille de l'ensemble $\{(i_1, \cdots, i_n) | 1 \leq i_j \leq 2 \text{ et } y = m_1^{i_1} m_2^{i_2} \cdots m_n^{i_n}\}$. En d'autres termes, $\#_D(x, y)$ est le nombre de façons par lesquelles on peut obtenir y à partir de x en dupliquant des messages. Comme dans les cas des pertes de messages, on définit :

$$P_D(x, y) \stackrel{\text{def}}{=} \#_D(x, y) \cdot \lambda_D^{|y|-|x|} \cdot (1 - \lambda_D)^{|x|},$$

qui est étendu aux contenus de canaux par $P_D(\mathbf{w}_1, \mathbf{w}_2) = \prod_{c \in \mathsf{C}} P_D(\mathbf{w}_1(c), \mathbf{w}_2(c))$.

On peut vérifier que pour $x \in \mathsf{M}^*$, $\sum_{y \in \mathsf{M}^*} P_D(x, y) = 1$. En effet, de façon similaire au cas des pertes, $\sum_{y \in M^k} \#_D(x, y) = \binom{k}{k-|x|}$, puisque cela revient à choisir parmi k messages, les $k-|x|$ qui ont été dupliqués pour obtenir à partir de x un mot de longueur k. Or $\binom{k}{k-|x|} = \binom{k}{|x|}$. Alors :

$$\begin{aligned}
\sum_{y \in \mathsf{M}^*} P_D(x, y) &= \sum_{y \in \mathsf{M}^*} \#_D(x, y)\ \lambda_D^{|y|-|x|}\ (1 - \lambda_D)^{|x|} \\
&= \sum_{k \in \mathbb{N}} \sum_{y \in \mathsf{M}^k} \#_D(x, y)\ \lambda_D^{k-|x|}\ (1 - \lambda_D)^{|x|} \\
&= \sum_{k \in \mathbb{N}} \lambda_D^{k-|x|}\ (1 - \lambda_D)^{|x|} \sum_{y \in \mathsf{M}^k} \#_D(x, y) \\
&= \sum_{k \in \mathbb{N}} \lambda_D^{k-|x|}\ (1 - \lambda_D)^{|x|} \binom{k}{|x|} \\
&= 1 \qquad\qquad \text{par la formule du binôme.}
\end{aligned}$$

On obtient une chaîne de Markov (Conf, P'_D) où $\mathsf{Conf} = (S \times (\mathsf{C} \to \mathsf{M}^*))$ comme avant et

$$P'_D((s_1, \mathbf{w}_1)(s_2, \mathbf{w}_2)) = \sum_{\mathbf{w}_3 \in (\mathsf{C} \to \mathsf{M}^*)} P((s_1, \mathbf{w}_1), (s_2, \mathbf{w}_3)) \cdot P_D(\mathbf{w}_3, \mathbf{w}_2). \tag{3.2}$$

Remarque 3.10. *La définition de P'_D revient à considérer que les duplications ont lieu après les actions et les pertes. D'autre part, on suppose qu'un message peut être dupliqué au plus une fois à chaque étape. Des remarques similaires pourront être faites lors de l'étude des LCS avec insertion ou corruption (voir les paragraphes suivants).*

Tous ces choix sont faits en vue de simplifier les calculs et les définitions. Bien sûr des variantes sont possibles. Dans tous les cas qui nous sont venus à l'esprit, les résultats de décidabilité présentés ci-dessous pourraient être adaptés.

Lemme 3.11. *Soit $\langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w, \lambda_D \rangle$ un PLCS avec erreurs de duplication, tel que $\lambda \geq \lambda_D > 0$. Alors, l'ensemble $Q_0 = \{(s, \varepsilon) | s \in S\}$ est un attracteur.*

Preuve : Regardons ce qui arrive à un message m fixé lors des phases de duplication et de perte. Avec probabilité $(1 - \lambda_D)$, la duplication n'affectera pas le message m, et il sera alors perdu avec probabilité λ. Sinon, avec probabilité λ_D, le message sera dupliqué et ses deux copies peuvent être perdues avec probabilité λ^2. Ces deux comportements sont ceux possibles pour qu'un message soit perdu. Ainsi, la probabilité qu'un message soit effacé est $p_{\text{eff}} = \lambda - \lambda\lambda_D + \lambda^2\lambda_D$.

De façon similaire, on exprime la probabilité qu'un message soit dupliqué et que chacune de ses deux copies ne soit pas affectée par les pertes : $p_{\text{aug}} = \lambda_D(1-\lambda)^2 = \lambda_D - 2\lambda\lambda_D + \lambda^2\lambda_D$.

Si $\lambda > \frac{\lambda_D}{1+\lambda_D}$, alors $p_{\text{eff}} > p_{\text{aug}} > 0$. C'est en particulier le cas quand $\lambda \geq \lambda_D > 0$. En reprenant la notion de niveau pour les PLCS (le niveau i est composé des configurations de taille i), à partir d'une configuration de niveau i, l'espérance du niveau des successeurs est inférieure à i. Ainsi la chaîne de Markov induite par le PLCS est orientée à gauche et le Théorème 3.2 s'applique. L'ensemble S_0 des configurations avec canal vide est un attracteur (fini) pour $\mathcal{M}_{\mathcal{P}}$ quand $\mathcal{P}$ est un PLCS avec erreurs de duplications tel que $\lambda \geq \lambda_D$. ☐

Les lemmes 3.9 et 3.11 entraînent le résultat suivant :

Théorème 3.12. *Les problèmes d'accessibilité probabiliste et d'accessibilité répétée probabiliste sont décidables pour les PLCS avec erreurs de duplication lorsque $\lambda \geq \lambda_D > 0$.*

Corruption On considère maintenant des LCS avec erreurs de corruption : lorsqu'ils sont dans les canaux, les messages peuvent être corrompus et transformés en d'autres messages, ceci en plus des pertes. On supposera que $|\mathsf{M}| > 1$ pour que les corruptions puissent modifier quelque chose au comportement normal.

Comme on vient de le voir pour les LCS avec erreurs et duplications, on étend la sémantique des LCS pour tenir compte des corruptions possibles.

Soit $x \in \mathsf{M}^*$ un contenu de canal. On note $C(x)$ l'ensemble $\{y \in \mathsf{M}^* || y| = |x|\}$. Ainsi, les éléments de $C(x)$ sont obtenus à partir de x si certains messages sont corrompus. Cette définition est étendue aux configurations. De plus, on ajoute à la relation de transition du LCS la règle suivante :

– Si $(s_1, \mathbf{w}_1) \to (s_2, \mathbf{w}_2)$ alors $(s_1, \mathbf{w}_1) \to (s'_2, \mathbf{w}'_2)$ pour tout $(s'_2, \mathbf{w}'_2) \in C((s_2, \mathbf{w}_2))$.

La décidabilité de l'accessibilité dans les LCS avec erreurs de corruption vient du fait que $(s_1, w_1) \xrightarrow{+} (s_2, w_2)$ implique $(s_1, w_1) \xrightarrow{+} (s_2, w_3)$ pour tout w_3 tel que $|w_3(c)| = |w_2(c)|$, pour chaque canal $c \in \mathsf{C}$. Ainsi, le seul élément à retenir dans l'analyse d'accessibilité est le nombre de messages présents dans chaque canal. Le problème est donc réduit à un problème d'accessibilité dans un LCS pour lequel l'alphabet des messages est un singleton. Le problème de l'accessibilité contrainte peut être résolu de la même façon.

Lemme 3.13. *Soit $\mathcal{L}$ un LCS avec des erreurs de corruption.*

1. *Pour toutes configurations s_1 et s_2, on peut décider si s_2 est accessible à partir de s_1. Le graphe $Graph(A)$ est donc calculable pour un ensemble fini de configurations donné A.*
2. *Pour toute configuration s et tout ensemble $Q \subseteq S$ on peut décider si l'ensemble des configurations (q, w) avec $q \in Q$ est accessible à partir de s.*

Un PLCS avec erreurs de corruption est de la forme $\langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w, \lambda_C\rangle$ où $\langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w\rangle$ est un PLCS et $\lambda_C \in [0, 1]$, représente la probabilité qu'un message soit transformé en un autre.

Soient $x, y \in \mathsf{M}^*$. On définit $\#_C(x, y)$ comme étant le cardinal l'ensemble $\{i|x(i) \neq y(i)\}$, c'est-à-dire le nombre d'éléments qui doivent être corrompus pour passer de x à y. Les probabilités associées sont définies comme suit :

$$P_C(x,y) = \begin{cases} (\frac{\lambda_C}{|\mathsf{M}|-1})^{\#_C(x,y)} \cdot (1-\lambda_C)^{|x|-\#_C(x,y)} & \text{si } |x| = |y|, \\ 0 & \text{sinon.} \end{cases} \tag{3.3}$$

De cette façon, $P_C(x, y)$ est la probabilité que x devienne y quand $\#_C(x, y)$ de ses lettres sont corrompues de façon indépendante avec probabilité λ_C.

Remarquons que pour $k \leq |x|$, il y a exactement $(|\mathsf{M}| - 1)^k\binom{|x|}{k}$ mots pour lesquels $\#_C(x, y) = k$. Alors on vérifie que $\sum_{y \in \mathsf{M}^* P_C(x,y)=1}$.

La fonction P_C est étendue de M^* à Conf de la manière habituelle. On peut alors définir la chaîne de Markov induite par le PLCS avec des erreurs de corruption : $\mathcal{M} = (\mathsf{Conf}, P'_C)$ où

$$P'_C((s_1, \mathbf{w}_1), (s_2, \mathbf{w}_2)) = \sum_{\mathbf{w}_3 \in (\mathsf{C} \to \mathsf{M}^*)} P((s_1, \mathbf{w}_1), (s_2, \mathbf{w}_3)) \cdot P_C(\mathbf{w}_3, \mathbf{w}_2). \tag{3.4}$$

Lemme 3.14. *Soit $\langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w, \lambda_C\rangle$ un PLCS avec erreurs de corruption, tel que $\lambda > 0$. Alors l'ensemble $Q_0 = \{(s, \varepsilon)|s \in S\}$ est un attracteur.*

Preuve : Les corruptions ne changent pas le nombre de messages dans le canaux. Alors la même preuve que pour les PLCS (avec pertes seules) permet de conclure que Q_0 est un attracteur. □

Les lemmes 3.13 et 3.14 permettent de conclure :

Théorème 3.15. *Les problèmes d'accessibilité probabiliste et d'accessibilité répétée probabiliste sont décidables pour les PLCS avec erreurs de corruption.*

Insertion On s'intéresse maintenant aux LCS qui, en plus des pertes, ont des erreurs d'insertion : c'est-à-dire pour lesquels des messages arbitraires peuvent apparaître dans les canaux. Ce modèle est introduit pas Cécé *et al.* dans [CFP96]. Comme précédemment, on étend la sémantique des LCS pour prendre en compte les insertions.

Pour $x \in \mathsf{M}^*$ et par analogie avec $D(x)$, on définit $I(x)$ comme l'ensemble $\{y \in \mathsf{M}^*|x \sqsubseteq y\}$ des sur-mots de x. En fait $I(x) = \uparrow x$, et l'on peut étendre la définition de I à $\mathsf{C} \to \mathsf{M}^*$ et Conf, comme on l'a expliqué pour $\uparrow$.

La relation de transition des LCS est alors augmentée de la règle suivante :

– Si $(s_1, \mathbf{w}_1) \to (s_2, \mathbf{w}_2)$ alors $(s_1, \mathbf{w}_1) \to (s'_2, \mathbf{w}'_2)$ pour tout $(s'_2, \mathbf{w}'_2) \in I((s_2, \mathbf{w}_2))$.

Le problème de l'accessibilité dans les LCS avec erreurs d'insertion est décidable [CFP96]. Ce résultat repose sur le fait que

$$(s_1, \mathbf{w}_1) \to (s_2, \mathbf{w}_2) \iff (s_1, \mathbf{w}_1) \to (s_2, \varepsilon).$$

Un PLCS avec erreurs d'insertion est un n-uplet $\langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w, \lambda_I \rangle$ où $\lambda_I \in [0, 1[$ régit la probabilité qu'un message soit inséré. Plus précisément, on suppose que l'insertion de messages suit une loi géométrique : la probabilité que k messages soient insérés à une étape est par définition $\lambda_I^k \cdot (1 - \lambda_I)$. D'autres choix seraient possibles.

La définition de $P_I(x, y)$, probabilité que x devienne y après insertion de messages, distingue plusieurs cas. On pose $P_I(x, x) = (1 - \lambda_I)$ et, si $|y| > |x|$, on définit $P_I(x, y)$ par induction sur le nombre de messages insérés, *i.e.*, $|y| - |x|$:

$$P_I(x, y) = \lambda_I \sum_{z \in \mathsf{M}^{|x|+1}} \frac{\#(x, z)}{|\mathsf{M}| \cdot (1 + |x|)} \cdot P_I(z, y). \tag{3.5}$$

Dans tous les autres cas, on pose $\mathcal{P}_I(x, y) = 0$.

On utilise alors l'égalité combinatoire qui suit :

$$\sum_{z \in \mathsf{M}^{|x|+1}} \#(x, z) = |\mathsf{M}| \cdot (1 + |x|), \tag{3.6}$$

pour montrer par récurrence sur k que

$$\sum_{y \in \mathsf{M}^{|x|+k}} P_I(x, y) = \lambda_I^k (1 - \lambda_I). \tag{3.7}$$

Une conséquence directe est que $\sum_{y \in \mathsf{M}^*} P_I(x, y) = 1$ pour tout $x \in \mathsf{M}^*$.

Comme on l'a fait pour P_D et P_C, on étend P_I de M^* à Conf. Le PLCS avec erreurs d'insertion induit une chaîne de Markov (S, P'_I) avec

$$P'_I((s_1, \mathbf{w}_1)(s_2, \mathbf{w}_2)) = \sum_{\mathbf{w}_3 \in (\mathsf{C} \to \mathsf{M}^*)} P((s_1, \mathbf{w}_1), (s_2, \mathbf{w}_3)) \cdot P_I(\mathbf{w}_3, \mathbf{w}_2). \tag{3.8}$$

La distribution que nous avons choisie fait que la probabilité que k messages soient introduits ne dépend pas de la taille et du contenu des canaux, ce que l'on constate dans l'équation 3.7. En conséquence, le système sera, comme c'est le cas des PLCS classiques, attiré vers les configurations à peu de messages.

Lemme 3.16. *Soit $\langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w, \lambda_I \rangle$ un PLCS avec erreurs d'insertion, tel que $\lambda > 0$. Alors l'ensemble $Q_0 = \{(s, \varepsilon) | s \in S\}$ est un attracteur.*

Preuve : Soit K l'espérance du nombre de messages insérés en une étape. Alors l'espérance du prochain niveau, étant donnée une configuration σ de taille n, vérifie :

$$\mathbb{E}(\sigma) \leq (n + 1)(1 - \lambda) + K$$

Alors, pour $\delta > 0$ fixé, en choisissant $N_0 \in \mathbb{N}$ suffisamment grand, $\mathbb{E}(\sigma) < n$ pour tout indice $n \geq N_0$, et la chaîne de Markov est presque orientée à gauche. Le Théorème 3.5 (cf. page 62) s'applique et Q_0 est un attracteur. $\square$

Comme les problèmes d'accessibilité et d'accessibilité contrainte sont décidables pour les PLCS avec erreur d'insertion, le lemme 3.16 entraîne :

Théorème 3.17. *Les problèmes d'accessibilité probabiliste et d'accessibilité répétée probabiliste sont décidables pour les PLCS avec erreurs d'insertion.*

3.5.2 Autres types d'erreurs

L'approche que nous avons développée pour les PLCS avec duplications, corruption et insertion, peut être adapté à des PLCS où plusieurs sources de non-fiabilité sont combinées. Par exemple, on pourrait s'intéresser à des modèles où des erreurs de duplication et de corruption s'ajoutent aux pertes. Dans tous les cas, nos méthodes s'adaptent si l'accessibilité reste décidable et s'il existe un attracteur fini. Pour l'existence de l'attracteur fini, il est nécessaire que les erreurs qui font augmenter la taille des canaux (comme les duplications, les insertions) restent dominées par les pertes.

3.5.3 Model-checking

De la même façon que pour les PLCS, la vérification de formules du temps linéaire pour les PLCS avec erreurs du duplication, de corruption ou d'insertion est décidable, toujours sous réserve que l'attracteur fini existe. Par exemple, pour les PLCS avec erreurs de duplication, on impose (comme pour les problèmes d'accessibilité et d'accessibilité répétée) que $\lambda \geq \lambda_D > 0$.

L'idée essentielle est encore que l'attracteur fini permet d'analyser le comportement des chaînes de Markov induites par les PLCS en regardant les composantes fortement connexes finales.

Troisième partie

Lossy Channel Systems probabilistes et non-déterministes

Chapitre 4

Les LCS comme processus de décision markoviens

Sommaire

Les processus de décision markoviens permettent de représenter des systèmes combinant probabilités et non-déterminisme. Dans le cas des systèmes à canaux non fiables, la sémantique probabiliste pour le choix entre les actions, comme c'est le cas dans la deuxième partie, n'est pas pleinement satisfaisante. Certains comportements non-déterministes ne peuvent pas être représentés par des choix probabilistes, sans qu'ils soient dénaturés. Dans le cas de systèmes distribués par exemple, l'ordre des actions entre les différents processus n'est pas établi à l'avance puisque l'évolution de chaque processus est indépendante des autres. Il est donc indispensable de conserver des choix non déterministes pour modéliser ces incertitudes. Un autre cas de non-déterminisme intrinsèque apparaît lorsque le système interagit avec un environnement dont le comportement n'est pas maîtrisé. En effet, là encore la connaissance imparfaite de l'environnement, et donc de ses actions, nécessite d'adopter un formalisme non déterministe. Enfin, on utilise souvent le non-déterminisme dans une première étape de conception, pour ne pas trancher entre les possibilités d'implémentation qui seront précisées dans une phase ultérieure. Pour toutes ces raisons, les probabilités ne peuvent pas se substituer au non-déterminisme dans bien des cas.

D'autre part, un inconvénient majeur de l'utilisation de lois de probabilité pour les choix entre les actions possibles pour les automates communicants est l'équité qu'elles engendrent. Plus précisément, si une transition de probabilité positive est tirable infiniment souvent, elle sera tirée presque sûrement infiniment souvent. Cette conséquence est fâcheuse dans le cas de systèmes purement non déterministes, pour lesquels le comportement normal pourrait tout à fait être inéquitable entre les règles.

Pour toutes ces raisons, il est commode de considérer un formalisme qui combine non-déterminisme et probabilités, raffiner l'étude de systèmes probabilistes et non déterministes.

Les *processus de décision markoviens* (dénotés PDM) constituent un tel formalisme. Les systèmes associés à ce modèle mathématique sont parfois aussi appelés *Probabilistic Nondeterministic Systems* [BA95].

4.1 Processus de décision markoviens

Soit E un ensemble dénombrable. On note $Dist(E)$ l'ensemble des distributions de probabilité sur E.

Définition 4.1 (Processus de décision markovien). *Un* processus de décision markovien *est un couple* $\texttt{PDM} = \langle E, \rightsquigarrow \rangle$ *où E est un ensemble dénombrable d'états et* $\rightsquigarrow : E \rightarrow 2^{Dist(E)}$ *associe à chaque état un ensemble fini de distributions sur S.*

Décrivons le comportement : étant donné un état $x \in E$, on peut considérer que les successeurs sont choisis en deux phases. Supposons que $x \rightsquigarrow \{p_1^x, \cdots, p_{k_x}^x\}$. Tout d'abord, une distribution p_i^x est choisie de façon non déterministe parmi $p_1^x, \cdots, p_{k_x}^x$, puis un successeur est choisi en fonction de la distribution p_i^x.

Dans la littérature, on trouve d'autres définitions pour les processus de décisions markoviens. Par exemple, il est possible de considérer que l'espace des configurations est une partition (S_n, S_p) entre les configurations « non déterministes » et les configurations « probabilistes ». Dans cette approche, les transitions sont décrites par deux fonctions T et p : T associe à chaque configuration de S_n un ensemble de successeurs et p associe à chaque configuration de S_p une distribution sur S.

Nous adoptons dans cette thèse le formalisme choisi par Bianco et de Alfaro dans [BA95] pour les Probabilistic Nondeterministic Systems. D'autres appellations des processus de décisions markoviens se trouvent dans la littérature : *chaînes de Markov réactives* [Var99] ou *chaînes de Markov concurrentes* [Var85, HSP83].

Remarque 4.2. *Dans cette thèse, comme c'était déjà le cas pour les chaînes de Markov, on ne s'intéressera qu'à des processus de décision markoviens à espace d'états dénombrable.*

4.1.1 Sémantique

Notion d'adversaire

Étant donné un processus de décision markovien, on ne peut pas définir directement un espace de probabilité sur les exécutions du système. On dispose néanmoins d'un espace de probabilité pour chaque arbre de choix non déterministes, c'est-à-dire une fois que sont fixés les choix non déterministes. Il est nécessaire de restreindre l'ensemble des chemins considérés en déterminant les choix entre les actions à chaque étape, pour pouvoir définir un espace de probabilité. De plus, la probabilité de passer d'une configuration à une autre dépend de l'action choisie : il se peut que deux transitions différentes fassent passer d'une même configuraton σ à une même configuration τ. Il est donc insuffisant de connaître la suite des configurations visitées le long d'un chemin fini pour calculer sa mesure.

On appelle *adversaire* le dispositif qui fait les choix non déterministes dans chaque configuration entre les différentes actions possibles. L'adversaire peut se baser sur l'historique des configurations visitées jusqu'alors. On utilise le terme *adversaire* pour ce qui est appelé *adversaire déterministe dépendant de l'historique*[1] dans la classification de Puterman [Put94].

[1] history-dependent deterministic scheduler

En effet, on ne s'intéresse dans cette thèse qu'à des adversaires déterministes (par opposition à *probabilistes*) : dans chaque configuration, en fonction des configurations déjà visitées, l'adversaire choisit un successeur plutôt qu'une distribution sur les successeurs. Nous justifierons ce choix plus tard (cf. Remarque 4.4).

Pour un adversaire donné, un processus de décision markovien induit une chaîne de Markov, ce qui permet de parler de la probabilité qu'une formule soit vérifiée, comme on l'a fait dans le cas des PLCS vus comme des chaînes de Markov (cf. Partie II). Partant d'un processus de décision markovien (à espace d'états dénombrable), la chaîne de Markov obtenue est encore à espace d'états dénombrable, même pour les adversaires les plus généraux possibles. Intuitivement, on obtient une chaîne de Markov, à partir d'un couple ($\mathtt{PDM}, \mathcal{U}$) (processus de décision markovien et adversaire), en dépliant $\mathtt{PDM}$ en un arbre ayant $\mathtt{S}^+$ comme feuilles et en élaguant les branches qui ne suivent pas les choix de $\mathcal{U}$. Une définition formelle est donnée par la suite.

Vérification Introduisons à présent des questions de vérification qui se posent pour les processus de décision markoviens. On montre comment, ces questions combinent celles des systèmes probabilistes et celles des systèmes non déterministes. Dans le cas de la vérification de formules du temps linéaire pour les systèmes non déterministes, on dit qu'un modèle satisfait une spécification donnée, si pour toute exécution du système, la propriété est vraie. Pour les chaînes de Markov, modèle naturel des systèmes purement probabilistes, on se demande, étant donnée une formule, si la probabilité de l'ensemble des chemins satisfaisant la spécification est égale à 1 ou non. Le cas des processus de décision markoviens mixe les deux approches puisque la question naturelle qui se pose est la suivante : pour tout adversaire, la chaîne de Markov obtenue satisfait-elle la spécification presque sûrement ?

Plus généralement, on considérera les questions suivantes : quel que soit l'adversaire, la probabilité que la formule ϕ soit vérifiée est-elle 1 ? est-elle positive ? ou encore, existe-t-il un adversaire tel que la probabilité que la formule ϕ soit vérifiée est 1 ? est positive ?

Il est fréquemment plus aisé de raisonner en utilisant la formulation existentielle pour les adversaires : « existe-t-il un adversaire tel que... ». Nous considérerons donc la plupart du temps les questions sous cette forme (plutôt que sous la forme universelle) car elle est plus naturelle pour poser des problèmes de vérification. On s'intéressera donc aux quatre problèmes de vérification qualitative : existe-t-il un adversaire tel que la chaîne de Markov induite vérifie la propriété

- presque sûrement (*i.e.*, avec probabilité 1) ?
- avec probabilité strictement positive ?
- avec probabilité < 1 ?
- avec probabilité nulle ?

Définissons formellement la notion d'adversaire pour un processus de décision markovien.

Définition 4.3 (Adversaire)**.** *Un adversaire pour $\mathcal{N}$ est une application $\mathcal{U}$ qui associe à chaque chemin fini π dans $\mathcal{N}$, une règle de transition $\delta \in \Delta$ qui est tirable dans la dernière configuration de π.*

On trouve d'autres appellations dans la littérature : *policy* dans [dA99], *scheduler* pour les systèmes distribués [PZ86], ou encore *strategy*.

Remarque 4.4. *Les adversaires que nous considérons sont déterministes (aussi appelés markoviens) : pour chaque chemin fini, ils choisissent une règle de transition à appliquer. Plus* *généralement on peut définir des adversaires stochastiques qui associent à un chemin fini une distribution sur les transitions possibles dans la configuration courante. Le cas des adversaires déterministes est donc un cas particulier.*

Dans de nombreux cadres, on restreint l'étude aux adversaires déterministes puisqu'ils suffisent à atteindre les extremums (minimum et maximum) de la probabilité qu'une certaine propriété soit vérifiée. En conséquence, la vérification qualitative peut se limiter à la classe des adversaires déterministes. C'est par exemple le cas de la vérification de formules des logiques pCTL et pCTL pour les processus de décision markoviens à espace d'état dénombrable [BA95].*

Définition 4.5 (Exécution conforme)**.** *Soit $\mathcal{N}$ un processus de décision markovien et $\mathcal{U}$ un adversaire pour $\mathcal{N}$. Une exécution $\Pi : \sigma_0 \xrightarrow{\delta_0} \sigma_1 \xrightarrow{\delta_1} \sigma_2 \cdots$ est dite* conforme à $\mathcal{U}$ *si pour tout préfixe $\pi = \sigma_0 \xrightarrow{\delta_0} \sigma_1 \cdots \sigma_{n-1} \xrightarrow{\delta_{n-1}} \sigma_n$ de Π, $\mathcal{U}(\pi) = \delta_n$.*

Les exécutions conformes constituent exactement l'ensemble des exécutions dans la chaîne de Markov engendrée par le processus de décision markovien et l'adversaire $\mathcal{U}$ fixé. Étant donné un adversaire $\mathcal{U}$, on note $\mathbb{P}_\mathcal{U}$ la probabilité définie par $\mathcal{U}$ sur le processus de décision markovien. Ainsi, $\mathbb{P}_\mathcal{U}(\sigma \models \phi)$ dénote la mesure de l'ensemble des exécutions issues de σ, conformes à $\mathcal{U}$ et vérifiant la propriété ϕ.

De façon similaire, on parlera de *chemin conforme* à $\mathcal{U}$ pour les chemins finis $\Pi : \sigma_0 \xrightarrow{\delta_0} \sigma_1 \cdots \sigma_n$ qui respectent les choix de l'adversaire $\mathcal{U}$.

Chaîne de Markov induite La combinaison d'un processus de décision markovien `PDM` et d'un adversaire $\mathcal{U}$ donne lieu à une chaîne de Markov $\texttt{PDM}_\mathcal{U} = \langle S^+, P_\mathcal{U} \rangle$. Pour tout $x \in S^+$ suite finie d'états, on définit :

$$P_\mathcal{U}(x\sigma, x\sigma\sigma') \stackrel{\text{def}}{=} \begin{cases} P(\sigma, \sigma') & \text{si } \mathcal{U}(x\sigma) = \delta \text{ et } \sigma \xrightarrow{\delta} \sigma', \\ 0 & \text{sinon.} \end{cases}$$

et $P_\mathcal{U}(x\sigma, y\sigma') = 0$ lorsque $y \neq x\sigma$.

Par la suite, nous introduisons quelques sous-classes d'adversaires qui correspondent à des hypothèses sur l'environnement : les adversaires à mémoire finie, sans mémoire, ou aveugles. La description d'un adversaire dans ces classes est plus simple que celle des adversaires généraux ; de plus ces classes restreintes possèdent des propriétés intéressantes vis-à-vis de la vérification de propriétés de vivacité par exemple.

Adversaires à mémoire finie et sans mémoire

Nous introduisons les adversaires à mémoire finie. Ces adversaires modélisent fidèlement le comportement d'un système réel implémenté. Ils sont basés sur un ensemble fini de *modes* qui guident leur choix dans chaque configuration. Ainsi pour une configuration donnée, les choix de l'adversaire peuvent être différents selon le mode, mais les adversaires à mémoire finie ne sont pas aussi puissants que les adversaire généraux, qui peuvent être vus comme des adversaires possédant un nombre dénombrable de modes. La définition formelle d'un adversaire à mémoire finie est donnée ci-dessous.

Définition 4.6. *Un adversaire* à mémoire finie *pour $\mathcal{N}$ est un adversaire $\mathcal{U}$ pour lequel il existe un quadruplet (U, D, η, u_0) tel que pour toute configuration σ_0, si $\pi = \sigma_0 \to \cdots \to \sigma_n$, alors*

$$\mathcal{U}(\pi) = D(u, \sigma_n) \qquad \text{où } U \ni u = \eta^*(u_0, \pi) \stackrel{\text{def}}{=} \eta(\cdots \eta(\eta(u_0, \sigma_0), \sigma_1, \cdots, \sigma_n).$$

Dans cette définition, U est un ensemble fini de *modes* et $u_0 \in U$ le mode initial. La fonction $D : U \times \mathsf{Conf} \to \Delta$ est la *règle de décision* qui associe à chaque paire (u, σ) une règle $\delta \in \Delta(\sigma)$, et $\eta : U \times \mathsf{Conf} \to U$ est une application qui détermine le prochain mode.

Les modes peuvent être utilisés pour stocker une information de taille bornée sur le chemin parcouru jusqu'alors. On peut, par exemple, utiliser un adversaire fini pour rendre équitables les choix entre un nombre fixé de transitions : si $\{\delta_0, \cdots, \delta_k\} \subseteq \Delta(\sigma)$ sont des règles de transition tirables en σ, et que l'on veut qu'elles soient tirées tour à tour quand σ est visitée, il suffit de prendre k modes $0, 1, \cdots, k-1$ qui dénotent chacun la règle qui sera tirée lors de la prochaine visite de σ. Dans cet exemple, la fonction de changement de mode est donnée par $\eta(i, \sigma) = i + 1 \mod k$ et la décision de $\mathcal{U}$ en σ par $D(i, \sigma) = \delta_i$.

Remarque 4.7. *Un adversaire à mémoire finie ne peut pas être en général représenté finiment puisque l'ensemble des configurations est potentiellement infini et que les choix de l'adversaire n'ont,* a priori, *aucune propriété de régularité. En particulier, D et η ne sont pas des fonctions récursives en général.*

Nous introduisons une sous-classe des adversaires à mémoire finie, les adversaires sans mémoire.

Définition 4.8 (Adversaire sans mémoire)**.** *Un adversaire à mémoire finie $\mathcal{U} = (U, D, \eta, u_0)$ est* sans mémoire *s'il ne possède qu'un seul mode : $|U| = 1$.*

Un adversaire sans mémoire fait toujours le même choix, dans une configuration donnée ; ses choix ne dépendent pas de l'historique des configurations déjà visitées. On peut alors le donner sous la forme plus simple d'une application de l'ensemble des configurations dans l'ensemble des règles de transitions $\mathcal{U} : \mathsf{Conf} \to \Delta$.

4.2 Cas des NPLCS

Dans cette section, nous détaillons la sémantique des processus de décision markoviens pour les LCS. Nous avons introduit la notion de NPLCS -Nondeterministic Probabilistic Channel Systems- dans [BS03]. Ces derniers permettent de modéliser le choix entre les actions de l'automate de façon non déterministe et les pertes de messages de façon probabiliste.

4.2.1 Sémantique opérationnelle

Soit $\mathcal{L}$ un LCS, et λ un taux de perte. Pour toute configuration $\sigma \in \mathsf{Conf}$, on dénote par $Dist(\sigma)$ la distribution définie par

$$Dist(\sigma)(\sigma') = P_L(\sigma, \sigma')$$

où P_L est la probabilité de passer de σ à σ' en perdant des messages (définie formellement dans la Partie II page 43).

Le non-déterminisme du choix entre les actions et le caractère aléatoire des pertes de messages sont combinés de la façon suivante.

Définition 4.9. $\texttt{PDM}_{\mathcal{L}} = \langle \mathsf{Conf}, \rightsquigarrow \rangle$, *le processus de décision markovien associé à* $\mathcal{L}$ *est donné par : Si* $\sigma \xrightarrow{\delta}_{\text{perf}} \tau$, *alors* $\sigma \xrightarrow{\delta} Dist(\tau)$ *est une règle de transition de* $\texttt{PDM}_{\mathcal{L}}$.

Ainsi, à chaque configuration σ, la relation de transition $\rightsquigarrow$ associe un ensemble de distributions, une pour chaque successeur parfait de σ dans $\mathcal{L}$. Le comportement de $\texttt{PDM}_{\mathcal{L}}$ est une succession de choix non déterministes de la prochaine étape entrecoupés de pertes probabilistes.

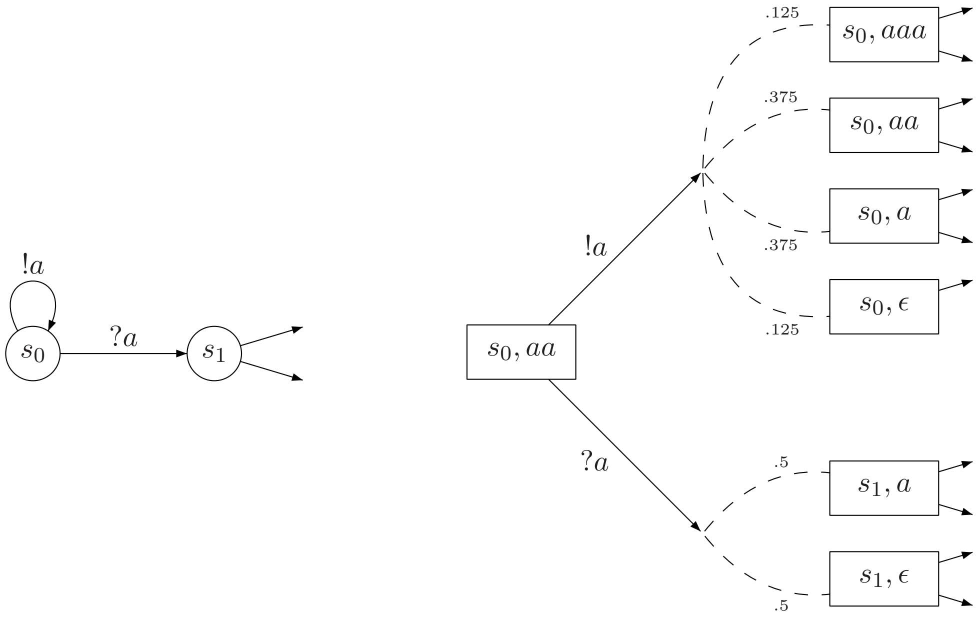

FIG. 4.1 – La sémantique des NPLCS sur un exemple

Exemple 4.10. *La sémantique des NPLCS est illustrée Figure 4.1. Dans la partie gauche sont représentés deux états et deux transitions d'un automate communicant. En ajoutant un taux de pertes* λ, *ce dernier peut être vu comme un NPLCS* $\mathcal{N}$. *Sur l'exemple, on a choisi* $\lambda = \frac{1}{2}$. *On représente à droite de la figure, une partie du processus de décision markovien associé à* $\mathcal{N}$, *avec pour configuration initiale* (s_0, aa). *À partir de cette configuration, deux règles de transition sont tirables :* $\Delta(s_0, aa) = \{(s_0, ?a, s_1), (s_0, !a, s_0)\}$. *Ces deux transitions engendrent chacune une distribution de probabilité représentée en pointillés, selon les pertes qui ont lieu après avoir effectué la transition choisie.*

Une fois défini le processus de décision markovien $\texttt{PDM}_{\mathcal{L}}$ à partir de $\mathcal{L}$, on peut considérer les adversaires d'un NPLCS. L'adversaire fait le choix des actions (lecture, écriture, action interne) tandis que les pertes sont résolues de façon probabiliste.

4.2.2 Attracteur fini

Soit $\mathcal{N} = \langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda \rangle$ un NPLCS fixé. Étant donné un adversaire $\mathcal{U}$, le processus de décision markovien associé à $\mathcal{N}$ donne une chaîne de Markov, notée $\mathcal{M}_{\mathcal{N}}^{\mathcal{U}}$. Dans cette chaîne de Markov, comme c'était le cas pour $\mathcal{M}_{\mathcal{P}}$, chaîne de Markov d'un PLCS $\mathcal{P}$, l'ensemble des configurations où les canaux son vides constitue un attracteur fini. Si $\mathcal{U}$ est un adversaire pour $\mathcal{N}$ et T un ensemble de configurations, on dira que T est un attracteur pour $\mathcal{U}$ si $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond T) = 1$ pour toute configuration σ. Cette terminologie est moins lourde que de parler de l'attracteur de la chaîne de Markov obtenue à partir de $\texttt{PDM}_{\mathcal{N}}$ en fixant l'adversaire $\mathcal{U}$.

Théorème 4.11 (Attracteur fini)**.** *Pour tout adversaire $\mathcal{U}$, l'ensemble $Q_0 \stackrel{\text{def}}{=} \{(s,\varepsilon) | s \in A\}$ est un attracteur pour $\mathcal{U}$.*

Preuve : Soit $\mathcal{U}$ un adversaire pour $\mathcal{N}$. Alors $\mathcal{M}_{\mathcal{N}}^{\mathcal{U}}$ est une chaîne de Markov. De la même façon que pour les chaînes de Markov engendrées par les PLCS, $\mathcal{M}_{\mathcal{N}}^{\mathcal{U}}$ est presque orientée à gauche. Les états de la chaîne de Markov $\mathcal{M}_{\mathcal{N}}^{\mathcal{U}}$ ne sont plus simplement des configurations $\sigma = (s, \mathbf{w})$, mais contiennent l'historique des configurations visitées jusqu'alors. On les regroupe néanmoins toujours en fonction de la taille de la configuration courante. Pour cette partition la même démonstration que pour le Lemme 3.6 (cf. page 64) permet de montrer que $\mathcal{M}_{\mathcal{N}}^{\mathcal{U}}$ est presque orientée à gauche. $\square$

Nous donnons également une propriété des adversaires à mémoire finie pour les NPLCS.

Proposition 4.12. *Soient $\mathcal{U}$ un adversaire à mémoire finie et $\sigma \in \mathsf{Conf}$ une configuration.*

1. *Si $\tau \in \mathsf{Conf}$, u est un mode de $\mathcal{U}$ et $S(\tau_u)$ l'ensemble des configurations qui sont accessibles depuis τ_u par $\mathcal{U}$ alors :*

$$\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond\tau_u) = \Pr_{\mathcal{U}}(\sigma \models \bigwedge_{\tau' \in S(\tau_u)} \Box\Diamond\tau').$$

2. *Si $Q \subseteq S$ est un ensemble d'états de contrôle,*

$$\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond Q) = \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond Q_\varepsilon)$$

où $Q_\varepsilon \stackrel{\text{def}}{=} \{(s,\varepsilon) | s \in Q\}$.

Preuve : Le premier point repose sur le Lemme 2.4 (cf. page 42). En effet si $\tau' \in S(\tau_u)$ alors τ' est accessible depuis τ_u par un chemin conforme à $\mathcal{U}$. Dans la chaîne de Markov $\mathcal{M}_{\mathcal{N}}^{\mathcal{U}}$, on peut donc appliquer le Lemme 2.4 à τ et τ'. Ceci est valable pour toutes les configurations de $S(\tau_u)$ (dont τ_u fait partie) ; ce qui donne le résultat.

Soit à présent $Q \subseteq S$ un ensemble d'états de contrôle. La deuxième assertion découle de 1. en utilisant le fait que si $(q, \mathbf{w})$ est accessible depuis σ en une ou plusieurs étapes, alors il en est de même de (q, ε). $\square$

Remarque 4.13. *Dans le cas restreint des adversaires sans mémoire, la Proposition 4.12 devient : soit τ une configuration, alors*

$$\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond\tau) = \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond\tau \wedge \bigwedge_{\tau' \in S(\tau)} \Box\Diamond\tau')$$

puisque $\mathcal{U}$ n'a qu'un mode.

4.2.3 Adversaires aveugles, presque aveugles

Dans le cadre des NPLCS, nous utiliserons des notions plus spécifiques pour les adversaires : aveugles, presque aveugles. Nous en donnons les définitions et les premières propriétés ci-dessous.

Définition 4.14 (Adversaire aveugle). *Un adversaire est* aveugle *si ses décisions ne dépendent que des états de contrôle, et pas du contenu des canaux.*

Formellement, si $\pi = \sigma_0 \to \sigma_1 \cdots \sigma_{n-1} \to \sigma_n$, avec $\sigma_i = (s_i, w_i)$, pour tout $\pi' = (s_0, w'_0) \to \cdots\cdots (s_n, w'_n)$ on a $\mathcal{U}(\pi) = \mathcal{U}(\pi')$.

Comme les choix probabilistes affectent seulement les pertes, et donc uniquement les contenus de canaux, les projections sur les états de contrôle des exécutions conformes à un adversaire aveugle sont toutes identiques.

Remarque 4.15. *Notons qu'un adversaire aveugle ne choisit jamais de transitions qui sont des lectures. En effet, il doit choisir une action qui aurait aussi bien pu être tirée dans le même état de contrôle avec tous les canaux vides. Néanmoins, ce type d'adversaires sera utile par la suite.*

Un adversaire est appelé presque aveugle s'il se comporte à partir d'un moment comme un adversaire aveugle. Plus précisément, nous donnons ci-dessous la définition formelle.

Définition 4.16 (Adversaire presque aveugle). *Un adversaire $\mathcal{U}$ est* presque aveugle *s'il existe un adversaire $\mathcal{W}$ et un adversaire aveugle $\mathcal{U}$ tels que pour toute configuration σ et presque toute exécution conforme à $\mathcal{U}$ partant de σ : $\pi = \sigma = \sigma_1 \to \sigma_2 \to \cdots$, il existe un entier $n \geq 0$ vérifiant :*

- $\mathcal{U}(\sigma_1 \to \cdots \sigma_i) = \mathcal{W}(\sigma_1 \to \cdots \sigma_i)$ *pour tout* $i \leq n$, *et*
- $\mathcal{U}(\sigma_1 \to \cdots \sigma_i) = \mathcal{V}(\sigma_1 \to \cdots \sigma_i)$ *pour tout* $i > n$.

Chapitre 5

Vérification qualitative

Sommaire

Dans ce chapitre nous étudions des questions de vérification qualitative pour les NPLCS en nous intéressant tout d'abord aux adversaires les plus généraux possibles. On montre alors l'indécidabilité de la vérification qualitative de propriétés du temps linéaire. Cela contraste avec les résultats obtenus dans la section 2.3 pour les PLCS (cf. page 47). La source de l'indécidabilité réside dans le pouvoir des adversaires. Dans un deuxième temps, nous nous restreignons donc à une classe d'adversaires, celle des adversaires à mémoire finie, pour laquelle on prouve la décidabilité de la vérification qualitative des formules LTL. Enfin, nous montrerons comment généraliser les résultats obtenus dans le cas d'adversaires équitables (à mémoire finie). Une caractéristique importante dans l'étude des propriétés qualitatives des NPLCS est que le taux de perte λ n'a pas d'influence sur le caractère décidable des problèmes, ni sur les propriétés qualitatives vérifiées. Par exemple, comme c'était le cas pour les PLCS (cf. Partie II), on peut prouver l'existence d'un attracteur fini quel que soit le taux de perte $\lambda > 0$.

Remarque 5.1. *Nous avons introduit les LCS probabilistes et non déterministes (NPLCS) dans [BS03], et poursuivi leur étude dans [BS04]. Dans ces premiers travaux, les problèmes étaient légèrement différents, puisque à la fois les adversaires utilisés mais aussi les propriétés considérées n'étaient pas les mêmes. Tout d'abord, on considérait des adversaires qui pouvaient choisir entre rester dans un état de contrôle ou tirer une transition. Cette capacité de*

« idling » s'est révélée trop puissante par la suite, notamment lorsque l'on s'interessa à des questions de vivacité. D'autre part, nous ne considérions dans ce travail préliminaire que des formules dont les propositions atomiques étaient des ensembles d'états de contrôle. Il était alors impossible d'exprimer des propriétés concernant les contenus des canaux. Notre étude a ensuite été étendue dans ces deux directions [BBS07].

5.1 Indécidabilité de la vérification qualitative

Le résultat principal de l'étude des NPLCS avec adversaires généraux est le suivant. Étant donnée une propriété exprimée sous forme d'une formule de logique temporelle ϕ, les problèmes de vérification qualitative rappelés ci-dessous sont indécidables, en général (même dans le cas de formules relativement simples) :

– Existe-t-il un adversaire $\mathcal{U}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \phi) = 1$?
– Existe-t-il un adversaire $\mathcal{U}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \phi) > 0$?

Par dualité, c'est aussi le cas pour les questions « avec probabilité < 1 » et avec probabilité nulle.

Nous montrons ces résultats d'indécidabilité pour des cas particuliers de formules LTL, aussi simples que possible. On montrera dans la section 5.2 que ces formules sont les plus petites pour lesquelles on a l'indécidabilité.

5.1.1 Gadget de nettoyage

Dans les preuves d'indécidabilité (et aussi celles de difficulté - voir la sous-section 5.2.4), nous utiliserons un gadget de nettoyage qui permet de vider les canaux de communications d'un LCS, sans introduire de blocage. Le gadget de nettoyage est un NPLCS représenté Figure 5.1. Il peut être intégré à un NPLCS plus grand, symbolisé ici par les flèches en pointillé (une qui arrive sur `in` et une qui sort de `out`). Pour simplifier, nous considérons ici le cas d'un canal unique, il n'est donc pas précisé dans les règles de transitions.

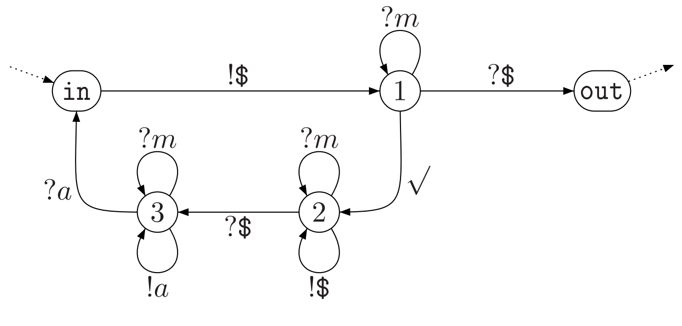

FIG. 5.1 – Gadget de nettoyage $\mathsf{Net}_{\mathsf{M}}$, avec $\$ \notin \mathsf{M}$

Étant donné un alphabet M, le NPLCS de la Figure 5.1 utilise un nouveau symbole $\$ \notin \mathsf{M}$. La lettre a est une lettre $a \in \mathsf{M}$ choisie une fois pour toutes, et les boucles de lecture $?m$ concernent les $|\mathsf{M}| + 1$ messages du nouvel alphabet $\mathsf{M} \cup \{\$\}$. Le nouveau message $\$$ permet de vérifier que le canal est bien vide pour passer de 1 à `out`. Dans la suite on note $\mathsf{Net}_{\mathsf{M}}$ le LCS constituant le gadget de nettoyage paramétré par l'alphabet de messages M.

Nous allons décrire l'ensemble des configurations accessibles du gadget. Soit $T \subseteq \mathsf{Conf}$ la région dénotée par l'expression régulière suivante :

$$T = (\mathtt{in}, \mathsf{M}^*) + (1, \mathsf{M}^*(\$ + \varepsilon)) + (2, \mathsf{M}^*\$^*) + (3, \$^*\mathsf{M}^*) + (\mathtt{out}, \varepsilon).$$

Lemme 5.2. *T coïncide avec $Post^*(\mathtt{in}, \mathsf{M}^*)$.*

Preuve : On prouve que l'ensemble des configurations accessibles depuis la région $(\mathtt{in}, \mathsf{M}^*)$ est inclus dans T en montrant que T est un invariant par rapport aux transitions du gadget. Par exemple, en appliquant la transition $(2, ?\$, 3)$ sur la région $(2, \mathsf{M}^*\$^*)$, on obtient la région $(3, \$^*)$ qui est inclus dans T (dans $(3, \$^*\mathsf{M}^*)$ plus précisément).

L'inclusion inverse est montrée en exhibant, pour chaque configuration $\sigma \in T$, une exécution particulière menant à σ. Remarquons qu'on peut choisir des exécutions sans perte. Par exemple, la configuration $(2, w\$^k)$ est atteignable depuis $(\mathtt{in}, w)$ en écrivant $\$$ pour rejoindre $(1, w\$)$, puis en tirant la transition $\surd$ pour atteindre $(2, w\$)$. □

Les propriétés de Net_M dont nous aurons besoin par la suite sont listées ci-dessous :

Lemme 5.3. *Pour tout $w \in \mathsf{M}^*$,*

1. *Si $\mathcal{U}$ est un adversaire et $v \neq \varepsilon$ alors*

$$\mathrm{Pr}_\mathcal{U}((\mathtt{in}, w) \models \Diamond(\mathtt{out}, v)) = 0.$$

2. *Il existe un adversaire (sans mémoire) $\mathcal{U}$ pour lequel*

$$\mathrm{Pr}_\mathcal{U}((\mathtt{in}, w) \models \Diamond(\mathtt{out}, \varepsilon)) = 1.$$

Preuve : La première propriété peut être renforcée : en effet, l'ensemble des exécutions partant de $(\mathtt{in}, w)$ et visitant $(\mathtt{out}, v)$ avec $v \neq \varepsilon$ est non seulement de mesure nulle, mais plus précisément vide, par le Lemme 5.2.

La seconde propriété exprime qu'il existe un adversaire qui parviendra presque sûrement à vider le canal en parcourant le gadget depuis `in` jusqu'à `out`. On montre de plus que cet adversaire peut être choisi sans mémoire. Dans l'état de contrôle 1, si le symbole $\$$ est en tête du canal, $\mathcal{U}$ choisit de lire ce $\$$ et de passer dans l'état `out`. Sinon, il choisit de lire les messages de M tant que le canal commence par $m \in \mathsf{M}$. Si le canal est vide, il choisit de passer dans l'état de contrôle 2. Dans l'état de contrôle 2, $\mathcal{U}$ écrit $\$$ dans le canal jusqu'à ce que ce symbole soit conservé et peut ensuite tirer la transition de lecture $(2, ?\$, 3)$. Dans l'état 3, $\mathcal{U}$ écrit une lettre de l'alphabet originel $a \in \mathsf{M}$ et la lit dès que celle-ci n'est pas perdue pour atteindre $(\mathtt{in}, \varepsilon)$. En répétant ces opérations, presque sûrement, le $\$$ écrit entre les états `in` et 1 sera conservé et pourra être lu pour rejoindre $(\mathtt{out}, \varepsilon)$. En effet, il y a une probabilité strictement positive que le symbole $\$$ ne soit pas perdu. L'ensemble des configurations accessibles depuis $(\mathtt{in}, \mathsf{M}^*)$ par des exécutions conformes à $\mathcal{U}$ est exactement :

$$T_0 = (\mathtt{in}, \mathsf{M}^*) + (1, \mathsf{M}^*(\$ + \varepsilon)) + (2, \$ + \varepsilon) + (3, \mathsf{M} + \varepsilon) + (\mathtt{out}, \varepsilon).$$

À partir de la deuxième visite de l'état initial, l'ensemble des configurations visitées se restreint même à :

$$T_0' = (\mathtt{in}, \varepsilon) + (1, \$ + \varepsilon) + (2, \$ + \varepsilon) + (3, \mathsf{M} + \varepsilon) + (\mathtt{out}, \varepsilon).$$

□

Remarque 5.4. *Il est impossible de construire un gadget vérifiant les deux propriétés du Lemme 5.3 sans introduire un nouveau symbole (un message qui n'est pas dans* M*).*

Plus précisément, supposons que partant de $\sigma_0 = (s_0, w_0)$*, avec* $w_0 \in \mathsf{M}^*$ *et* $s_0 = \mathtt{in}$*, on puisse atteindre par un chemin* π *une configuration finale* $(\mathtt{out}, \varepsilon)$ *dans un NPLCS* ayant pour alphabet M. *On définit une suite de configurations* (σ_i) *extraite du chemin* π*. La configuration* $\sigma_{i+1} = (s_{i+1}, w_{i+1})$ *est celle obtenue lorsque toutes les lettres de* w_i *ont été lues ou perdues, c'est-à-dire lorsque le mot* w_i *n'est plus dans le canal. La suite* (σ_i) *est finie car* $\sigma_0 \xrightarrow{*} (\mathtt{out}, \varepsilon)$ *et le dernier élément de la suite est* $(\mathtt{out}, \varepsilon) = (s_{n+1}, w_{n+1})$*. On a donc l'exécution suivante* $\sigma_0 \xrightarrow{\rho_1} \sigma_1 \cdots \xrightarrow{\rho_{n+1}} \sigma_{n+1} = (\mathtt{out}, \varepsilon)$*, où les* ρ_i *sont des suites de transitions a priori avec pertes. Notons que tous les contenus du canal (i.e. les* (w_i)*) sont des mots sur l'alphabet* M*. Pour tout mot* $u \in \mathsf{M}^*$*, l'exécution suivante est valide :*

$$(s_0, w_0 w_1 \cdots w_n u) \xrightarrow{\rho_1} (s_1, w_1 \cdots w_n u w_1) \cdots \xrightarrow{\rho_{n+1}} (s_{n+1}, u w_1 \cdots w_n) = (\mathtt{out}, u w_1 \cdots w_n)$$

Alors, le NPLCS ne vérifie pas l'assertion 1. *du Lemme 5.3.*

5.1.2 Un cas d'indécidabilité de l'accessibilité répétée

Nous utilisons à présent le gadget de nettoyage intégré à un NPLCS arbitraire pour prouver l'indécidabilité de l'existence d'un adversaire pour lequel une propriété de Büchi est vérifiée avec probabilité positive. Plus précisément, nous réduisons le problème de la finitude (*boundedness*) pour montrer l'indécidabilité. Rappelons qu'un LCS $\mathcal{L}$ est *borné* pour une configuration initiale donnée si l'ensemble des configurations accessibles depuis cette configuration initiale est fini. Le problème de la finitude est défini ainsi :

Finitude
Instance : Un LCS $\mathcal{L}$, une configuration $\sigma \in \mathsf{Conf}$.
Question : L'ensemble $\mathit{Post}^*(\sigma)$ des configurations atteignables depuis σ dans $TS_{\mathcal{L}}$ est-il fini ?

Richard Mayr a prouvé l'indécidabilité de ce problème :

Théorème 5.5 ([May03])**.** *Le problème de la finitude est indécidable pour les LCS.*

Nous utilisons ce résultat en réduisant le problème de la finitude à une question d'accessibilité répétée. Plus précisément, étant donné un LCS quelconque $\mathcal{L}$, on construit un NPLCS $\mathcal{N}' \stackrel{\text{def}}{=} \langle \mathcal{L}', \lambda \rangle$ tel que $\mathcal{L}$ est borné si et seulement si il existe un adversaire pour $\mathcal{N}'$ tel que la probabilité de visiter un état donnée infiniment souvent est positive.

Définissons formellement le problème :

Accessibilité répétée avec probabilité positive
Instance : Un NPLCS $\mathcal{N} = (S, \mathsf{C}, \mathsf{M}, \Delta, \lambda)$, un ensemble d'états de contrôle $A \subseteq S$, une configuration initiale σ.
Question : Existe-t-il un adversaire $\mathcal{U}$ satisfaisant $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A) > 0$?

Théorème 5.6. *Le problème de l'accessibilité répétée avec probabilité positive est indécidable.*

Preuve : Soit $\mathcal{L}$ un LCS ayant un unique canal $c \in \mathsf{C}$ et une configuration initiale fixée $\sigma_0 = (s_0, \varepsilon)$. Ceci n'est pas une restriction. En effet, le résultat de Mayr (cf. Théorème 5.5) est encore valable avec ces hypothèses.

On modifie $\mathcal{L}$ en ajoutant d'une part le gadget de nettoyage et, d'autre part, trois états de contrôle supplémentaires `rejouer`, `succes` et `puits`. On ajoute ensuite les règles de transition permettant de passer de tout état r de $\mathcal{L}$ à `rejouer` et `succes` par des actions internes. Depuis l'état `succes`, `rejouer` est accessible via une transition de lecture, alors qu'une action interne permet de passer dans `puits` qui ne peut pas être quitté. À partir de l'état `rejouer` on peut entrer dans le gadget de nettoyage afin de revenir en (s_0, ε). La construction de $\mathcal{L}'$, le nouveau LCS, à partir de $\mathcal{L}$ est représentée Figure 5.2, page 85.

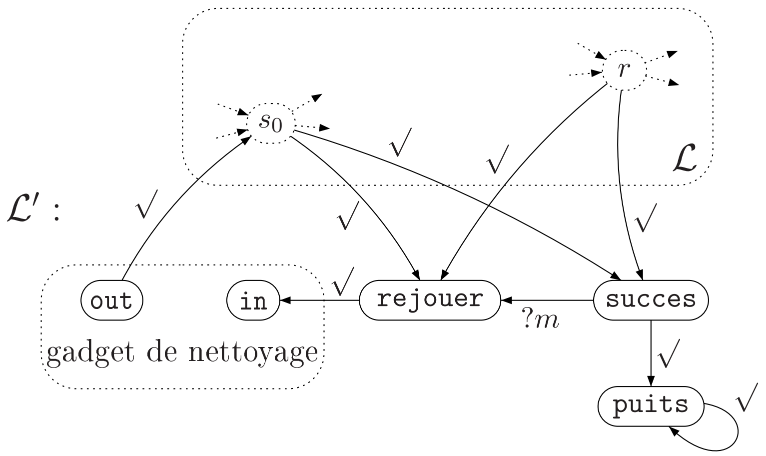

FIG. 5.2 – Le LCS $\mathcal{L}'$ associé à $\mathcal{L}$ dans la preuve du Théoreme 5.6

On note $\mathcal{N}'$ le NPLCS obtenu à partir de $\mathcal{L}'$ en lui adjoignant un taux de perte quelconque λ : $\mathcal{N}' = \langle \mathcal{L}', \lambda \rangle$. Le gadget de nettoyage permet de revenir à la configuration initiale de $\mathcal{L}$. Ainsi, toute exécution dans $\mathcal{L}'$ est une suite d'exécutions dans $\mathcal{L}$ entrecoupées de visites aux « nouveaux » états.

On montre, dans un premier temps, que si $\mathcal{L}$ est borné en partant de (s_0, ε) alors, il n'existe pas d'adversaire garantissant $\Box\Diamond$`succes` avec probabilité positive.

Lemme 5.7. *Supposons que $\langle \mathcal{L}, \sigma_0 \rangle$ est borné. Alors pour tout adversaire $\mathcal{U}$ pour $\mathcal{N}'$,* $\Pr_{\mathcal{U}}(\sigma_0 \models \Box\Diamond\texttt{succes}) = 0$.

Preuve : Soit $\mathcal{U}$ un adversaire quelconque pour $\mathcal{N}'$. On considère les exécutions conformes à $\mathcal{U}$ qui visitent l'état `succes` infiniment souvent ; ces exécutions forment un ensemble Π. Soit $\pi \in \Pi$ une telle exécution dans $\mathcal{L}'$. L'état `succes` n'est accessible que depuis un état du LCS originel $\mathcal{L}$. Ainsi, si π visite `succes` infiniment souvent, π passe infiniment souvent de $\mathcal{L}$ à `succes`. Puisque $\mathcal{L}$ est borné depuis (s_0, ε), et que chaque retour à $\mathcal{L}$ commence en (s_0, ε), il existe une configuration $\tau = (s, w)$ à partir de laquelle on passe infiniment souvent dans `succes`. En (s, w), la probabilité de perdre tous les messages en tirant la transition vers `succes` est $\lambda^{|\sigma|}$. En conséquence, avec probabilité 1, la configuration $(\texttt{succes}, \varepsilon)$ sera atteinte.

Plus précisément, on peut partitionner les exécutions de Π selon l'ensemble des configurations `Inf` du LCS $\mathcal{L}$ qu'elles visitent infiniment souvent avant de visiter `succes` :

$$\begin{aligned}
\Pr_{\mathcal{U}}(\Pi) &= \Pr_{\mathcal{U}}(\Pi \wedge \Box\Diamond\mathsf{Conf}_{\mathcal{L}}) \\
&= \Pr_{\mathcal{U}}\big(\Pi \wedge \bigvee_{\mathtt{Inf}\subseteq\mathsf{Conf}_{\mathcal{L}}} \bigwedge_{\tau\in\mathtt{Inf}} \Box\Diamond(\tau \wedge \bigcirc\mathtt{succes})\big) \\
&= \Pr_{\mathcal{U}}\big(\Pi \wedge \bigvee_{\mathtt{Inf}\subseteq\mathsf{Conf}_{\mathcal{L}}} \bigwedge_{\tau\in\mathtt{Inf}} \Box\Diamond(\tau \wedge \bigcirc(\mathtt{succes},\varepsilon))\big) \\
&\leq \Pr_{\mathcal{U}}(\Box\Diamond(\mathtt{succes},\varepsilon)).
\end{aligned}$$

La dernière égalité vient du fait que la probabilité est non nulle à partir de τ de perdre tous les messages en allant dans `succes`. Enfin, l'inégalité repose sur :

$$\{\pi \mid \pi \models \Box\Diamond(\mathtt{succes},\varepsilon)\} = \bigcup_{\mathtt{Inf}\in\mathsf{Conf}_{\mathcal{L}}} \{\pi \mid \pi \models \Box\Diamond(\mathtt{succes},\varepsilon) \wedge \bigvee_{\mathtt{Inf}\in\mathsf{Conf}_{\mathcal{L}}} \bigwedge_{\tau\in\mathtt{Inf}} \Box\Diamond\tau\}.$$

À partir de la configuration $(\mathtt{succes},\varepsilon)$, seule la transition $\mathtt{succes} \xrightarrow{\checkmark} \mathtt{puits}$ est tirable. On en déduit $\Pr_{\mathcal{U}}(\Box\Diamond(\mathtt{succes},\varepsilon)) = 0$ puisque `succes` n'est pas atteignable depuis `puits`. Finalement, $\Pr_{\mathcal{U}}(\Pi) = 0$: l'ensemble des exécutions conformes à $\mathcal{U}$ qui vérifient $\Box\Diamond\mathtt{succes}$ est de mesure nulle. □

Réciproquement, on montre que si $\mathcal{L}$ n'est pas borné à partir de (s_0,ε), il existe un adversaire pour lequel `succes` est visité infiniment souvent avec probabilité positive.

Lemme 5.8. *Si $\mathcal{L}$ est non borné à partir de (s_0,ε), alors il existe un adversaire $\mathcal{U}$ pour $\mathcal{N}'$ tel que* $\Pr_{\mathcal{U}}(\sigma_0 \models \Box\Diamond\mathtt{succes}) > 0$.

Preuve : On construit ici un adversaire sous lequel `succes` est visité infiniment souvent avec probabilité positive. Puisque $\mathcal{L}$ est non borné à partir de σ_0, on peut définir une suite $(\sigma_n)_{n\in\mathbb{N}^*}$ de configurations accessibles depuis σ_0 telles que $|\sigma_n| \geq n$. On notera $\sigma_n = (s_n, w_n)$. L'adversaire est composé de phases numérotées $1, 2, \cdots$. Quand la n-ième phase commence, $\mathcal{U}$ est dans la configuration σ_0 et essaie d'atteindre σ_n. Ceci est possible car σ_n est accessible depuis σ_0 ; fixons π_n un chemin menant de σ_0 à σ_n, $\mathcal{U}$ cherche alors à suivre π_n. Néanmoins, $\mathcal{U}$ ne contrôle pas les pertes, et il se peut que l'exécution dévie du chemin π_n. Tant que les pertes sont celles espérées $\mathcal{U}$ continue à engendrer le chemin π_n. Lorsque les pertes ne sont pas les bonnes (trop ou pas assez, ou des messages différents), $\mathcal{U}$ décide d'abandonner temporairement et passe dans l'état `rejouer`, puis par le gadget de nettoyage pour retenter sa chance depuis σ_0. Il utilise ici la stratégie du Lemme 5.3 (cf. page 83). La probabilité que π_n soit suivi avec succès jusqu'à σ_n est strictement positive, alors presque sûrement, en essayant suffisamment longtemps, σ_n sera atteint. Quand σ_n est finalement atteint, $\mathcal{U}$ choisit de passer dans l'état `succes`, en prenant le risque de perdre tous les messages du canal (le mot w_n en entier). S'il reste au moins un message, $\mathcal{U}$ passe par `rejouer` et le gadget de nettoyage pour entamer la phase $n+1$. Si à un moment le canal est vide dans `succes`, $\mathcal{U}$ est forcé à rejoindre `puits`, et ne pourra plus visiter `succes`. Lors de ces phases successives, $\mathcal{U}$ tente de visiter infiniment souvent `succes`. Montrons à présent qu'il y parvient avec probabilité (strictement) positive.

Lors du saut de σ_n à `succes`, il y a une probabilité positive $P_L(w_n,\varepsilon) \leq \lambda^n$ que tous les messages soient perdus, menant à $(\mathtt{succes},\varepsilon)$. Si tel est le cas, $\mathcal{U}$ ne pourra jamais initier la phase $n+1$, et sera bloqué dans l'état `puits`. Néanmoins, la probabilité qu'un tel événement advienne lors d'une des phases et strictement inférieure à 1 puisque :

$$\begin{aligned}\mathrm{Pr}_{\mathcal{U}}(\sigma_0 \models \Box\Diamond\texttt{succes}) &= \prod_{n=1}^{\infty}(1 - P_L(w_n, \varepsilon)) \\ &\geq \prod_{n=1}^{\infty}(1-\lambda^n) \\ &> 0.\end{aligned}$$

$\square$

Remarque 5.9. *Le Lemme 5.8 peut être raffiné. En effet, si $\mathcal{L}$ n'est pas borné à partir de (s_0, ε), alors pour tout probabilité $p < 1$, il existe un adversaire $\mathcal{U}$ pour $\mathcal{N}'$ tel que*

$$\mathrm{Pr}_{\mathcal{U}}((s_0, \varepsilon) \models \Box\Diamond\texttt{succes}) > p.$$

Les deux lemmes précédents entraînent l'équivalence suivante :

Corollaire 5.10. *$\mathcal{L}$ est non borné (à partir de σ_0) si et seulement si il existe un adversaire $\mathcal{U}$ pour $\mathcal{N}' = \langle \mathcal{L}', \lambda \rangle$ tel que $\mathrm{Pr}_{\mathcal{U}}(\sigma_0 \models \Box\Diamond\texttt{succes}) > 0$.*

Ceci termine la preuve du Théorème 5.6 car la finitude des LCS est indécidable (cf. Théorème 5.5, page 84). $\square$

Remarque 5.11. *Il n'est pas nécessaire de considérer des ensembles de configurations compliqués pour prouver l'indécidabilité. En effet, le problème de l'accessibilité répétée ($\Box\Diamond A$) avec probabilité positive est indécidable dès que A est un ensemble d'états de contrôle, et même un singleton.*

Par dualité, on obtient le résultat suivant sur la persistance presque sûrement :

Théorème 5.12. *Soit $\mathcal{N}$ un NPLCS, $A \subseteq S$ un ensemble d'états de contrôle et $\sigma \in \mathsf{Conf}$ une configuration de $\mathcal{L}$. Le problème de l'existence d'un adversaire $\mathcal{U}$ pour lequel $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond\Box A) = 1$ est indécidable.*

Preuve : La preuve repose sur le Théorème 5.6 (cf. page 84) et la constatation suivante : pour tout adversaire $\mathcal{U}$, $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond\Box A) < 1$ si et seulement si $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond\overline{A}) > 0$. $\square$

Les Théorèmes 5.6 et 5.12 montrent que la vérification qualitative de formules LTL pour les questions

$$\exists\mathcal{U}\, \mathrm{Pr}_{\mathcal{U}}(\sigma_0 \models \phi) > 0 \quad \text{et} \quad \exists\mathcal{U}\, \mathrm{Pr}_{\mathcal{U}}(\sigma_0 \models \phi) < 1$$

sont indécidables. Il en est de même des deux autres variantes de la vérification qualitative comme on le montre dans ce qui suit.

5.1.3 Autres résultats d'indécidabilité

On considère la formule du temps linéaire : $\phi \stackrel{\text{def}}{=} \Box\Diamond B \wedge \Diamond\Box A$, où A et B sont des ensembles d'états de contrôle. Cette formule nous permet de montrer, par une variante de la réduction présentée pour le Théorème 5.6 (cf. page 84), que le problème $\exists\mathcal{U},\ \mathrm{Pr}_{\mathcal{U}}(\sigma \models \phi) = 1$ est indécidable. En utilisant la négation de la formule, on obtient également le résultat pour la vérification qualitative avec probabilité nulle. On verra dans la section consacrée à la décidabilité que pour les deux sous-formules de $\Box\Diamond B \wedge \Diamond\Box A$, et la vérification qualitative presque sûrement est décidable. La formule $\Box\Diamond B \wedge \Diamond\Box A$ est minimale en ce sens.

Lemme 5.13. *Le problème, étant donné un NPLCS $\mathcal{N}$, des ensembles d'états de contrôle $A, B \subseteq S$ et une configuration σ, de l'existence d'un adversaire $\mathcal{U}$ tel que*

$$\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond B \wedge \Diamond\Box A) = 1,$$

est indécidable.

Preuve : Cette preuve est similaire à celle du premier résultat d'indécidabilité (Théorème 5.6) ; à partir d'un LCS générique $\mathcal{L}$, on construit un NPLCS $\mathcal{N}'$ (avec un taux de perte arbitraire) utilisant le gadget de nettoyage, ayant 2 états de contrôle distingués, qui joueront le rôle des ensemble A et B de l'énoncé. L'existence d'un adversaire pour $\mathcal{N}'$ vérifiant la propriété $\Box\Diamond B \wedge \Diamond\Box A$ avec probabilité 1 sera équivalente à la non-finitude de $\mathcal{L}$. La réduction est représentée Fig. 5.3.

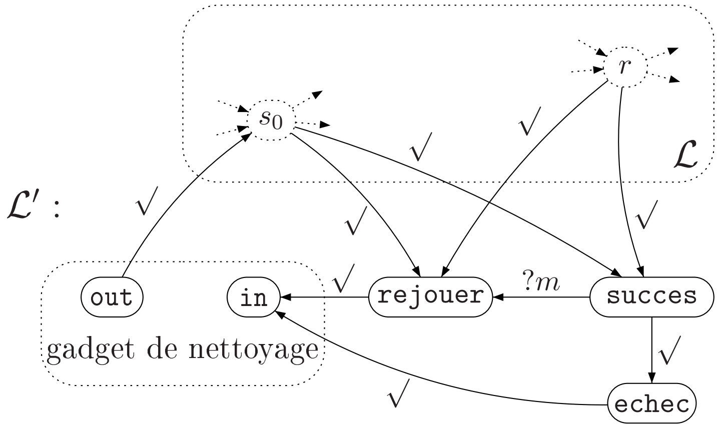

FIG. 5.3 – Le LCS $\mathcal{L}'$ associé à $\mathcal{L}$ dans la preuve du Lemme 5.13

Étant donné un LCS $\mathcal{L}$ avec un état initial distingué s_0, on l'intègre à un NPLCS $\mathcal{N}'$ en ajoutant trois nouveaux états de contrôle (`succes`, `rejouer` et `echec`) et le gadget de nettoyage. Plus précisément, de tout état de $\mathcal{L}$, on peut aller dans `rejouer` et `succes` par une transition sans lecture ni écriture. Depuis l'état `succes`, on atteint `rejouer` au moyen d'une lecture (il faut donc que le canal soit non vide pour y parvenir), ou `echec` par une transition ne touchant pas le contenu du canal. Enfin, depuis `rejouer` et `echec`, une transition mène à l'état initial du gadget de nettoyage. À la sortie du gadget de nettoyage, on entre à nouveau dans $\mathcal{L}$, plus précisément dans l'état initial s_0. Comme précédemment, on note $\mathcal{N}' = \langle\mathcal{L}', \lambda\rangle$ pour un $\lambda > 0$ arbitraire.

Proposition 5.14. *$\mathcal{L}$ est non borné (partant de (s_0, ε)) si et seulement si il existe un adversaire $\mathcal{U}$ pour $\mathcal{N}'$ tel que* $\mathrm{Pr}_{\mathcal{U}}((s_0, \varepsilon) \models \Box\Diamond\texttt{succes} \wedge \Diamond\Box\neg\texttt{echec}) = 1$.

Preuve : Si $\mathcal{L}$ n'est pas borné à partir de (s_0, ε), on construit un adversaire similaire à celui de la première preuve d'indécidabilité (cf. Théorème 5.6, page 84). L'adversaire chercher à quitter $\mathcal{L}$ pour atteindre `succes` avec un nombre de messages de plus en plus grand. La seule différence avec l'adversaire précédemment défini est son comportement hors de $\mathcal{L}$: cette fois-ci, si le canal est vide en entrant dans `succes` lors de la phase n, l'adversaire passe par `echec` puis par le gadget de nettoyage avant de tenter à nouveau d'atteindre σ_n. Ceci arrivera presque sûrement puisque $\sigma_0 \xrightarrow{*} \sigma_n$ et $\mathcal{U}$ passera ensuite à la phase $n+1$.

$$\mathrm{Pr}_{\mathcal{U}}\big((s_0, \varepsilon) \models \Box\Diamond\texttt{succes} \wedge \Diamond\Box\neg\texttt{echec}\big) = \lim_{n\to\infty} \prod_{k=n}^{\infty} (1 - \lambda^k).$$

En effet, $\prod_{k=n}^{\infty}(1-\lambda^k)$ est la probabilité qu'à partir de la phase n au moins un message soit conservé lors de chaque étape qui fait passer de $\mathcal{L}$ à `succes`. En prenant la limite lorsque n tend vers l'infini, on obtient la probabilité de visiter infiniment souvent `succes` et plus jamais `echec` à partir d'une certaine phase. Or, $\lim_{n\to\infty}\prod_{k=n}^{\infty}(1-\lambda^k) = 1$. On peut montrer ce dernier fait en utilisant des résultats sur les séries numériques.

En conclusion, $\mathrm{Pr}_{\mathcal{U}}\big((s_0, \varepsilon) \models \Box\Diamond\texttt{succes} \wedge \Diamond\Box\neg\texttt{echec}\big) = 1$.

Dans l'autre sens, soit $\mathcal{U}$ un adversaire quelconque. On va montrer que $\mathrm{Pr}_{\mathcal{U}}((s_0, \varepsilon) \models \Box\Diamond\texttt{succes} \wedge \Diamond\Box\neg\texttt{echec}) < 1$ si $\mathcal{L}$ est borné. À partir de maintenant, on parlera d'exécution pour « exécution conforme à $\mathcal{U}$ ». On considère l'ensemble des exécutions partant de σ_0 et visitant `succes` infiniment souvent. Cet ensemble peut être partitionné en fonction des configurations de $\mathcal{L}$ qui reviennent infiniment souvent juste avant `succes` au cours des exécutions. Soit Π un ensemble d'exécutions visitant infiniment souvent `succes` et ayant exactement `Inf` comme ensemble de configurations de $\mathcal{L}$ visitées infiniment souvent juste avant `succes` : $\Pi \stackrel{\text{def}}{=} \{\pi | \pi \models \bigwedge_{\tau\in\texttt{Inf}} \Box\Diamond(\tau \wedge \bigcirc\texttt{succes})\}$. Nous allons montrer que Π est de mesure nulle.

$$\begin{aligned}
\mathrm{Pr}_{\mathcal{U}}(\Pi) &= \mathrm{Pr}_{\mathcal{U}}(\sigma_0 \models \bigwedge_{\tau\in\texttt{Inf}} \Box\Diamond(\tau \wedge \bigcirc\texttt{succes})) \\
&= \mathrm{Pr}_{\mathcal{U}}(\sigma_0 \models \bigwedge_{\tau\in\texttt{Inf}} \Box\Diamond(\tau \wedge \bigcirc(\texttt{succes}, \varepsilon))) \qquad \text{car } \mathbf{P}(\tau, (\texttt{succes}, \varepsilon)) > 0 \\
&= \mathrm{Pr}_{\mathcal{U}}(\sigma_0 \models \bigwedge_{\tau\in\texttt{Inf}} \Box\Diamond(\tau \wedge \bigcirc(\texttt{succes}, \varepsilon) \wedge \bigcirc\bigcirc\,\texttt{echec}))
\end{aligned}$$

La dernière égalité vient du fait que `echec` est le seul état accessible à partir de la configuration $(\texttt{succes}, \varepsilon)$.

Ceci est vrai pour tout ensemble $\texttt{Inf} \subseteq \mathsf{Conf}_{\mathcal{L}}$, alors $\mathrm{Pr}_{\mathcal{U}}(\sigma_0 \models \Box\Diamond\texttt{succes} \wedge \Diamond\Box\neg\texttt{echec}) = 0$. On utilise ici le fait que $\mathcal{L}$ est borné et donc qu'il y a un nombre fini d'ensemble `Inf`. □

On conclut grâce au Théorème 5.5 (cf. page 84). □

Comme nous l'avons annoncé, le Théorème 5.6 et le Lemme 5.13 permettent de montrer que les quatre variantes de la vérification qualitative sont indécidables pour les formules du temps linéaire. Néanmoins, les formules pour lesquelles nous avons montré l'indécidabilité étaient différentes selon les cas. Il serait intéressant d'exhiber un fragment des formules du temps linéaire pour lequel les quatre variantes sont indécidables. Par exemple, à partir du Théorème 5.6 et du Lemme 5.13, on peut montrer l'indécidabilité de la vérification qualitative pour les formules de Streett, en choisissant judicieusement les ensembles d'états de contrôle.

Théorème 5.15 (Propriétés de Streett)**.** *Pour les propriétés qualitatives* $(a), \cdots, (d)$ *ci-dessous, le problème étant donné un NPLCS* $\mathcal{N}$*, une configuration* $\sigma \in \mathsf{Conf}$*, et* $2n$ *ensembles d'états* $A_1, B_1, \cdots, A_n, B_n \subseteq S$*, de l'existence d'un adversaire* $\mathcal{U}$ *tel que*

(a) $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i=1}^{n}(\Box\Diamond A_i \Rightarrow \Box\Diamond B_i)) > 0$,

(b) $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i=1}^{n}(\Box\Diamond A_i \Rightarrow \Box\Diamond B_i)) < 1$,

(c) $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i=1}^{n}(\Box\Diamond A_i \Rightarrow \Box\Diamond B_i)) = 1$,

(d) $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i=1}^{n}(\Box\Diamond A_i \Rightarrow \Box\Diamond B_i)) = 0$,

est indécidable.

Preuve : (a) découle du Théorème 5.6 puisque $\bigwedge_{i=1}^{n}(\Box\Diamond A_i \Rightarrow \Box\Diamond B_i)$ correspond à $\Box\Diamond B$ pour $n = 1$, $A_1 = S$ et $B_1 = B$.

(b) est même indécidable pour $n = 1$. En effet, pour $B = \emptyset$

$$\begin{aligned}
\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A \Rightarrow \Box\Diamond B) < 1 \text{ ssi } & \Pr_{\mathcal{U}}(\sigma \models \neg(\Box\Diamond A \Rightarrow \Box\Diamond B)) > 0 \\
\text{ssi } & \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A \wedge \Diamond\Box\overline{B}) > 0 \\
\text{ssi } & \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A) > 0
\end{aligned}$$

et on peut conclure avec le Théorème 5.6.

(c) est déduit du Lemme 5.13 avec $n = 2$, $A_1 = S$, $B_1 = B$, $A_2 = \overline{A}$ et $B_2 = \emptyset$ qui implique

$$\begin{aligned}
\bigwedge_{1 \le i \le n} (\Box\Diamond A_i \Rightarrow \Box\Diamond B_i) &\equiv (\Box\Diamond S \Rightarrow \Box\Diamond B) \wedge (\Box\Diamond\overline{A} \Rightarrow \Box\Diamond\emptyset) \\
&\equiv \Box\Diamond B \wedge \Diamond\Box A
\end{aligned}$$

(d) est indécidable dès le cas $n = 1$. Si $A, B \subseteq S$ sont des ensembles d'états de contrôle, on peut conclure grâce au Lemme 5.13 puisque :

$$\begin{aligned}
\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A \Rightarrow \Box\Diamond B) = 0 \text{ ssi } & \Pr_{\mathcal{U}}(\sigma \models \neg(\Box\Diamond A \Rightarrow \Box\Diamond B)) = 1 \\
\text{ssi } & \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A \wedge \Diamond\Box\overline{B}) = 1.
\end{aligned}$$

$\square$

Dans cette section, nous avons montré que pour certaines propriétés qualitatives, l'existence d'un adversaire est indécidable. En particulier, l'accessibilité répétée avec probabilité positive est un problème indécidable. C'est aussi le cas pour les propriétés que l'on peut exprimer par une formule de Streett, et ce quelles que soient les questions qualitatives (existence d'un adversaire avec probabilité nulle, positive, plus petite que 1, ou 1).

	$\mathbb{P}(\ldots) > 0$	$\mathbb{P}(\ldots) = 1$
$\Diamond A$	D	D
$\Box A$	D	D
$\bigwedge_i \Diamond A_i$	D	D
$\bigvee_i \Box A_i$	D	D
$\Box\Diamond A$	U	D
$\Diamond\Box A \wedge \Box\Diamond B$	U	U
$\bigwedge_i(\Box\Diamond A_i \to \Box\Diamond B_i)$	U	U

FIG. 5.4 – Table de résultats de décidabilité et indécidabilité

Remarque 5.16 (Classification des formules)**.** *À la fois pour le Théorème 5.6 et le Lemme 5.13, nous avons affirmé que les formules choisies étaient « petites » voire minimales. Pour hiérarchiser les formules, nous avons considéré la classification des formules du temps linéaire proposée par Chang, Manna et Pnueli [CMP92]. La formule $\Box\Diamond A$ exprime la persistance, et $\Box\Diamond B \wedge \Diamond\Box A$ combine persistance et réponse et est donc un cas particulier de réactivité. On montre dans la section qui vient que pour les propriétés des classes inférieures (sûreté et garantie) la vérification qualitative est décidable.*

5.2 Premiers résultats de décidabilité

Malgré les résultats d'indécidabilité présentés dans la section précédente, de nombreux problèmes sont décidables dans la vérification qualitative des NPLCS. Nous montrons en particulier la minimalité des formules proposées pour les cas d'indécidabilité.

Le tableau de la Figure 5.4 page 91 rappelle les résultats d'indécidabilité obtenus dans la section 5.1 et donne un aperçu des résultats de décidabilité que nous allons prouver.

Dans un premier temps, on considère les propriétés d'accessibilité (ou de façon duale, d'invariant) pour lesquelles on montre la décidabilité de la vérification qualitative.

5.2.1 Accessibilité

Nous nous intéressons aux propriétés d'accessibilité sous les quatre variantes de la vérification qualitative. En plus de la décidabilité, nous montrons que les adversaires à mémoire finie ont un pouvoir d'expression suffisant, c'est-à-dire que l'existence d'un adversaire vérifiant $\Diamond A$ avec probabilité $= 1$, $= 0$, > 0 ou < 1 est équivalente à l'existence d'un adversaire à mémoire finie ayant la même propriété.

On peut voir ici un parallèle avec les résultats de Bianco et de de Alfaro [BA95] qui ont démontré que lorsque que l'on considère uniquement des adversaires sans mémoire pour le model-checking de formules de la logique pCTL (CTL avec probabilités) des systèmes probabilistes et non déterministes, les bornes supérieures et inférieures des probabilités sont les mêmes que pour les adversaires les plus généraux possibles (avec mémoire et stochastiques).

Voici précisément les quatre problèmes que nous étudions :

accessibilité presque sûrement	$\exists\mathcal{U}\ \mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond A) = 1$?
accessibilité avec probabilité positive	$\exists\mathcal{U}\ \mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond A) > 0$?
invariant presque sûrement	$\exists\mathcal{U}\ \mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box A) = 1$?
invariant avec probabilité positive	$\exists\mathcal{U}\ \mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box A) > 0$?

Remarque 5.17. *Comme on l'a expliqué au chapitre précédent, par dualité, cela couvre aussi le traitement de* $\exists\mathcal{U}\ \mathrm{Pr}_{\mathcal{U}}(\cdots) < 1$ *et* $\exists\mathcal{U}\ \mathrm{Pr}_{\mathcal{U}}(\cdots) = 0$*.*

Lemme 5.18 (accessibilité, probabilité positive)**.** *Soit $\mathcal{N}$ un NPLCS, $\sigma \in \mathsf{Conf}$ une configuration et $A \subseteq \mathsf{Conf}$ un ensemble de configurations. Alors les propositions suivantes sont équivalentes :*

1. *il existe un adversaire $\mathcal{U}$ tel que $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond A) > 0$,*
2. *il existe un adversaire sans mémoire $\mathcal{U}$ tel que $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond A) > 0$,*
3. *$\sigma \xrightarrow{*} A$,*
4. *$\sigma \in \mu X.A \vee Pre(C_{\uparrow}X)$.*

Preuve : L'équivalence entre (3) et (4) repose sur l'écriture de $Pre^*(A)$ en mu-calcul et le fait que pour les LCS $Pre^*(X) = Pre(C_{\uparrow}X)$ (cf. Lemme 1.21, page 30).

(2) $\Rightarrow$ (1) est trivial.

(1) $\Rightarrow$ (3) est clair : s'il existe un adversaire $\mathcal{U}$ permettant d'atteindre A avec probabilité positive en partant de σ alors nécessairement A est accessible depuis σ dans le système de transition associé à $\mathcal{N}$.

(3) $\Rightarrow$ (2) : Soit π un chemin menant de σ à A : $\pi : \sigma = \sigma_0 \xrightarrow{\sigma}_1 \cdots \sigma_n \in A$. Sans perte de généralité, on peut supposer que σ_n est la première configuration de A rencontrée sur le chemin et que π est un chemin simple, *i.e.* qui visite au plus une fois chaque configuration. On définit un adversaire $\mathcal{U}$ qui vérifie l'assertion 2. L'adversaire $\mathcal{U}$ cherche à suivre le chemin π. Il y parvient avec probabilité positive puisque $\mathrm{Pr}(\pi) > 0$. Le fait que π soit simple permet de construire $\mathcal{U}$ sans mémoire. □

Grâce au Lemme 5.18 et au Théorème 1.26 (cf. page 32) sur le fragment gardé du μ-calcul, on déduit :

Corollaire 5.19. *Soient $\mathcal{N}$ un NPLCS et R une* région*. Alors l'ensemble des configurations $\sigma \in \mathsf{Conf}$ telles qu'il existe un adversaire $\mathcal{U}$ satisfaisant $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond R) > 0$ est une région effectivement calculable.*

Preuve : La preuve repose sur l'écriture de cet ensemble de configurations sous la forme $\mu X.R \vee Pre(C_{\uparrow}X)$ qui est un terme gardé. □

Pour la question de l'existence d'un adversaire $\mathcal{U}$ pour lequel $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box A) = 1$, nous introduisons une nouvelle notion d'accessibilité, que nous nommerons « *accessibilité de A sans risque par rapport à X* », où A et X sont deux ensembles de configurations.

Formellement,

Définition 5.20. *L'opérateur $\widehat{Pre}$: $\mathsf{Conf} \times \mathsf{Conf} \to \mathsf{Conf}$ est défini pour $A, X \subseteq \mathsf{Conf}$ deux ensembles de configurations de la façon suivante :*

$$\widehat{Pre}_X(A) \stackrel{\text{def}}{=} \bigcup_{\delta \in \Delta} Pre[\delta](A) \cap \widetilde{Pre}[\delta](X).$$

Si $\sigma \in \widehat{Pre}_X(A)$, il existe $\delta \in \Delta(\sigma)$ telle que $Post[\delta](\sigma) \subseteq X$ et $Post[\delta](\sigma) \cap A \neq \emptyset$. Autrement dit, si σ appartient à $\widehat{Pre}_X(A)$, on peut à partir de σ tirer une règle δ de façon à atteindre A avec probabilité positive, et en étant sûr de rester dans X.

Si $\sigma \in \widehat{Pre}_X(A)$, on dira que A *est accessible depuis* σ*, sans risque par rapport à* B.

Comme pour l'opérateur Pre, on définit par induction $\widehat{Pre}_X^{\leq n}(A)$ pour $n \in \mathbb{N}$: $\widehat{Pre}_X^{\leq 0}(A) = A$ et $\widehat{Pre}_X^{\leq n+1}(A) = \widehat{Pre}_X^{\leq n}(A) \cup \widehat{Pre}_X(\widehat{Pre}_X^{\leq n}(A))$. Enfin, $\widehat{Pre}_X^{*}(A) = \bigcup_{n\in\mathbb{N}} \widehat{Pre}_X^{\leq n}(A)$.

L'opérateur $\widehat{Pre}$ possède les caractéristiques suivantes :

Proposition 5.21. *Soient* A, B, X *des ensembles de configurations. Alors*
- $A \subseteq \widehat{Pre}_X^{*}(A)$;
- *si* $A \subseteq B$ *alors* $\widehat{Pre}_X^{*}(A) \subseteq \widehat{Pre}_X^{*}(B)$;
- $\widehat{Pre}_{\mathsf{Conf}}^{*}(A) = Pre^*(A)$.

On omet la preuve, qui est élémentaire.

De plus, en reprenant les notations de la section 1.3, l'opérateur $\widehat{Pre}$ vérifie la propriété :

Proposition 5.22. *Soient* A *et* X *des ensembles de configurations. Alors*
- $\widehat{Pre}_X(A) = \widehat{Pre}_{K_\downarrow X}(C_\uparrow A)$*, et*
- $\widehat{Pre}_X^{*}(A) = \widehat{Pre}_{K_\downarrow X}^{*}(C_\uparrow A)$.

Preuve : Il suffit de faire la preuve pour $\widehat{Pre}$, celle pour $\widehat{Pre}^{*}$ en découle. Soient $A, X \subseteq \mathsf{Conf}$. Pour montrer que $\widehat{Pre}_X(A) = \widehat{Pre}_{K_\downarrow X}(C_\uparrow A)$, on part de la définition de $\widehat{Pre}$, et on utilise le Lemme 1.21 (cf. page 30) :

$$\begin{aligned}
\widehat{Pre}_X(A) &= \bigcup_{\delta\in\Delta} Pre[\delta](A) \cap \widetilde{Pre}[\delta](X) && \text{par définition de } \widehat{Pre} \\
&= \bigcup_{\delta\in\Delta} Pre[\delta](C_\uparrow A) \cap \widetilde{Pre}[\delta](K_\downarrow X) && \text{grâce au Lemme 1.21} \\
&= \widehat{Pre}_{K_\downarrow X}(C_\uparrow A) && \text{par définition de } \widehat{Pre} \text{ à nouveau}
\end{aligned}$$

□

On peut maintenant énoncer une caractérisation de la vérification qualitative d'invariants presque sûrement.

Lemme 5.23 (invariant, probabilité 1)**.** *Soit* $\mathcal{N}$ *un NPLCS,* $\sigma \in \mathsf{Conf}$ *une configuration et* $A \subseteq \mathsf{Conf}$ *un ensemble de configurations. Les assertions suivantes sont équivalentes :*

1. *il existe un adversaire* $\mathcal{U}$ *tel que* $\Pr_{\mathcal{U}}(\sigma \models \Box A) = 1$,
2. *il existe un adversaire sans mémoire* $\mathcal{U}$ *tel que* $\Pr_{\mathcal{U}}(\sigma \models \Box A) = 1$,
3. $\sigma \in \nu X.A \wedge \widehat{Pre}_X(\mathsf{Conf})$.

Dans la suite, si ϕ est une formule de LTL, on notera $\mathbf{E}[\phi]$ l'ensemble des configurations σ telles qu'il existe un adversaire $\mathcal{U}$ pour lequel $\Pr_{\mathcal{U}}(\sigma \models \phi) = 1$. Les propriétés de $\mathcal{U}$ seront précisées dans le contexte (dans la section 5.3 par exemple, on se restreint à la classe des adversaires à mémoire finie).

Le lemme ci-dessus implique en particulier que $\mathbf{E}[\Box A] = \nu X.A \wedge \widehat{Pre}_X(\mathsf{Conf})$.

Preuve : Dans cette preuve on note $E \stackrel{\text{def}}{=} \nu X.A \wedge \widehat{Pre}_X(\mathsf{Conf})$.

(2) $\Rightarrow$ (1) : est trivial

(3) $\Rightarrow$ (2) : Supposons que $\sigma \in \nu X.A \wedge \widehat{Pre}_X(\mathsf{Conf})$ (*i.e.* $\sigma \in E$), et décrivons un adversaire sans mémoire qui assure $\Pr_{\mathcal{U}}(\sigma \models \Box A) = 1$. Pour tout $\tau \in E$ il existe une règle de transition δ_τ tirable en τ telle que $Post[\delta_\tau](\tau) \subseteq E$. L'adversaire $\mathcal{U}$ choisit dans une configuration τ la règle δ_τ, sans risque par rapport à E (quelques soient les pertes, on ne peut que rester dans E). Puisqu'il détermine la règle à tirer en fonction de la configuration courante seulement, $\mathcal{U}$ est sans mémoire. De plus, puisque $E \subseteq A$, toutes les exécutions engendrées par $\mathcal{U}$ à partir de σ satisfont $\Box A$; si bien que $\mathcal{U}$ satisfait sûrement (et *a fortiori* presque sûrement) $\Box A$.

(1) $\Rightarrow$ (3) : Soient $\sigma \in \mathsf{Conf}$ et $\mathcal{U}$ un adversaire vérifiant $\Pr_{\mathcal{U}}(\sigma \models \Box A) = 1$. On considère $T \stackrel{\text{def}}{=} \{\tau \in \mathsf{Conf} \mid \Pr_{\mathcal{U}}(\sigma \models \Diamond \tau) > 0\}$ l'ensemble des configurations qui peuvent être visitées le long d'une exécution conforme à $\mathcal{U}$. La configuration initiale σ appartient à T et $T \subseteq A$ (c'est l'hypothèse sur $\mathcal{U}$). De plus, tout chemin fini π de σ à $\tau \in T$ doit pouvoir être prolongé : ainsi, il existe une transition $\delta \in \Delta(\tau)$ que $\mathcal{U}$ peut être amené à choisir dans la configuration τ. Par définition de T, chaque configuration τ' telle que $\tau \xrightarrow{\delta} \tau'$ est dans T. Ainsi T est un post-point fixe de la fonctionnelle $F : X \rightarrow A \wedge \widehat{Pre}_X(\mathsf{Conf})$, c'est-à-dire $T \subseteq F(T)$. Puisque E est le plus grand point fixe de F, et donc l'union des post-points fixes, on conclut $\sigma \in T \subseteq E$. $\Box$

Remarque 5.24. *Le schéma de cette preuve, plus précisément de l'implication* (1) $\Rightarrow$ (3) *sera utilisé plusieurs fois dans cette thèse. Pour montrer qu'un ensemble* $\mathbf{E}[\phi]$ *est inclus dans un plus grand point fixe* $\nu X.F(X)$*, on procédera toujours ainsi. On fixe une configuration* $\sigma \in \mathbf{E}[\phi]$*. Alors, il existe un adversaire* $\mathcal{U}$ *tel que* $\Pr_{\mathcal{U}}(\sigma \models \phi) = 1$*. À partir de* $\mathcal{U}$*, on définit un ensemble de configurations* T *qui contient* σ*, et qui est un post-point fixe de* $F : T \subseteq F(T)$*. Ainsi,* T *est un sous-ensemble de* $\nu X.F(X)$*, le plus grand point fixe de* F*. Finalement* $\sigma \in T \subseteq \nu X.F(X)$*, ce qui permet de conclure.*

Remarque 5.25. *Remarquons également que l'adversaire que nous construisons dans la preuve de l'implication* (3) $\Rightarrow$ (2) *est tel que* $\sigma \models \Box A$*. Toutes les exécutions vérifient* $\Box A$*, ce qui est plus fort que* $\Pr_{\mathcal{U}}(\sigma \models \Box A) = 1$.

Ici encore, on utilise le Théorème 1.26 (cf. page 32) appliqué à l'algèbre des régions définie pour les LCS.

Corollaire 5.26. *Soient* $\mathcal{N}$ *un NPLCS et* R *une* région*. Alors l'ensemble des configurations* $\sigma \in \mathsf{Conf}$ *telles qu'il existe un adversaire* $\mathcal{U}$ *satisfaisant* $\Pr_{\mathcal{U}}(\sigma \models \Box R) = 1$ *est une région effectivement calculable.*

Preuve : D'après le Lemme 5.23, cet ensemble s'écrit $\nu X.R \wedge \widehat{Pre}_X(\mathsf{Conf})$ ou encore, grâce à la Proposition 5.22 (cf. page 93), $\nu X.R \wedge \widehat{Pre}_{K_\downarrow X}(\mathsf{Conf})$ qui est un terme gardé. $\Box$

Nous nous intéressons à présent au problème de l'invariant avec probabilité positive.

Lemme 5.27 (invariant, probabilité positive)**.** *Soit* $\mathcal{N}$ *un NPLCS,* $\sigma \in \mathsf{Conf}$ *une configuration et* $A \subseteq \mathsf{Conf}$ *un ensemble de configurations. Les assertions suivantes sont équivalentes :*

1. *il existe un adversaire* $\mathcal{U}$ *tel que* $\Pr_{\mathcal{U}}(\sigma \models \Box A) > 0$,
2. *il existe un adversaire sans mémoire* $\mathcal{U}$ *tel que* $\Pr_{\mathcal{U}}(\sigma \models \Box A) > 0$,
3. $\sigma \in \mu X.\mathbf{E}[\Box A] \vee (Pre(X) \wedge A)$.

Preuve : (2) $\Rightarrow$ (1) : est trivial

(3) $\Rightarrow$ (2) : Soit $\sigma \in E \stackrel{\text{def}}{=} \mu X.\mathbf{E}[\Box A] \vee (\mathit{Pre}(X) \wedge A)$. Cette formule peut se lire comme $\exists A$ *Until* $\mathbf{E}[\Box A]$. Alors, il existe un chemin fini partant de σ, ne visitant que des configurations de A et atteignant $\mathbf{E}[\Box A]$, que l'on note : $\pi : \ \sigma \xrightarrow{*}_A \sigma_n \in \mathbf{E}[\Box A]$. Comme précédemment, on suppose, sans perte de généralité, que π est simple et que σ_n est la première configuration de $\mathbf{E}[\Box A]$ visitée le long de π. Le comportement de $\mathcal{U}$ (qu'on est en train de définir) consiste à suivre le chemin π. Si σ_n est atteint, $\mathcal{U}$ se comporte alors comme l'adversaire décrit pour $\mathbf{E}[\Box A]$ (cf. Lemme 5.23, page 93). L'adversaire $\mathcal{U}$ ainsi construit est sans mémoire et satisfait $\Pr_{\mathcal{U}}(\sigma \models \Box A) > 0$.

(1) $\Rightarrow$ (3) : Soit $\mathcal{U}$ un adversaire vérifiant $\Pr_{\mathcal{U}}(\sigma \models \Box A) = 1$. Définissons $T \stackrel{\text{def}}{=} \{\tau \in \mathsf{Conf} \mid \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond\tau \wedge \Box A) > 0\}$. La propriété de l'attracteur fini (cf. Théorème 4.11, page 79) implique que T est non vide, et qu'il existe de plus un état de contrôle $s \in S$ tel que $(s, \varepsilon) \in T$. La non-vacuité de T implique l'existence d'un chemin $\sigma \xrightarrow{*}_A T$, c'est-à-dire d'un chemin qui part de σ ne visitant que des configurations de A et atteignant T. Il reste à montrer que $T \subseteq \mathbf{E}[\Box A]$. Nécessairement, $T \subseteq A$, sinon, l'ensemble des exécutions conformes à $\mathcal{U}$ et vérifiant $\Box A$ n'auraient pas une mesure positive. Pour $\tau \in T$ fixé, il existe un règle de transition $\tau \xrightarrow{\delta}$ telle que $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond\tau \wedge \Box\Diamond\text{“}\delta \text{ est tirée''} \wedge \Box A) > 0$ puisqu'il y a un nombre fini de règles tirables en τ. Si τ' est un successeur de τ par δ, $\mathbf{P}(\tau, \tau') > 0$ donc $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond\tau) = \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond\tau \wedge \Box\Diamond\tau')$. On en déduit

$$\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond\tau \wedge \bigwedge_{\tau' \in \delta(\tau)} \Box\Diamond\tau' \wedge \Box A) > 0.$$

Ainsi, $\tau \in \widehat{\mathit{Pre}}_T(\mathsf{Conf})$. Ce raisonnement vaut pour toute configurations $\tau \in T$, donc $T \subseteq \widehat{\mathit{Pre}}_T(\mathsf{Conf})$. Comme $\mathbf{E}[\Box A]$ est un plus grand point fixe, $T \subseteq \mathbf{E}[\Box A]$. En conclusion $\sigma \xrightarrow{*}_A (\mathbf{E}[\Box A])$. $\Box$

Corollaire 5.28. *Soient $\mathcal{N}$ un NPLCS et R une* région. *Alors, l'ensemble des configurations $\sigma \in \mathsf{Conf}$ telles qu'il existe un adversaire $\mathcal{U}$ satisfaisant $\Pr_{\mathcal{U}}(\sigma \models \Box R) > 0$ est une région effectivement calculable.*

Preuve : Cet ensemble s'écrit $\mu X.\mathbf{E}[\Box R] \vee (\mathit{Pre}(X) \wedge R)$ ou encore $\mu X.\mathbf{E}[\Box R] \vee (\mathit{Pre}(C_\uparrow X) \wedge R)$ par le Lemme 1.21 (cf. page 30). Or, on sait que $\mathbf{E}[\Box R]$ est calculable grâce au Corollaire 5.28. Le terme $\mu X.\mathbf{E}[\Box R] \vee (\mathit{Pre}(C_\uparrow X) \wedge R)$ étant gardé, le Théorème 1.26 (cf. page 32) s'applique. $\Box$

On en vient au problème de l'accessibilité presque sûrement.

Lemme 5.29 (accessibilité, probabilité 1)**.** *Soit $\mathcal{N}$ un NPLCS, $\sigma \in \mathsf{Conf}$ une configuration et $A \subseteq \mathsf{Conf}$ un ensemble de configurations. Alors les propositions suivantes sont équivalentes :*

1. *il existe un adversaire $\mathcal{U}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Diamond A) = 1$,*
2. *il existe un adversaire sans mémoire $\mathcal{U}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Diamond A) = 1$,*
3. *$\sigma \in \nu X.[\mu Y.(A \vee \widehat{\mathit{Pre}}_X(Y))] = \nu X.\widehat{\mathit{Pre}}^*_X(A)$.*

Autrement dit, avec les notations introduites précédemment, l'ensemble des configurations pour lesquelles il existe un attracteur assurant $\Box A$ presque sûrement s'écrit : $\mathbf{E}[\Diamond A] = \nu X.\widehat{\mathit{Pre}}^*_X(A)$.

Preuve : (2) $\Rightarrow$ (1) : est trivial

Dans le reste de la preuve, on note $E \stackrel{\text{def}}{=} \nu X.\widehat{Pre}^*_X(A)$.

(3) $\Rightarrow$ (2) : Montrons que si $\sigma \in E$ alors il existe un adversaire sans mémoire $\mathcal{U}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Diamond A) = 1$. Remarquons que si $\tau = (s, \mathbf{w}) \in E$, et $\tau \neq \sigma$, alors $(s, \varepsilon) \in E$. De plus, pour toute configuration $\sigma \in E$, il existe un chemin $\pi_\tau : \tau \stackrel{\text{def}}{=} \tau_0 \xrightarrow{\delta_1} \tau_1 \cdots \xrightarrow{\delta_n} \tau_n \in A$ tel que pour tout indice i, $Post[\delta_{i+1}](\tau_i) \subseteq E$. Le choix de ces chemins peut être fait de telle sorte que les chemins sont simples, et pour tout couple de configurations (τ, τ'), si τ' apparaît le long de π_τ, alors π'_τ est le suffixe de π_τ à partir de τ'. L'adversaire $\mathcal{U}$ que nous définissons est alors sans mémoire : dans une configuration τ, il choisit la première règle de transition δ_0 du chemin π_τ. Grâce à la propriété de l'attracteur fini, il existe une configuration (s, ε) visitée infiniment souvent presque sûrement. De plus, $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond(s, \varepsilon)) = \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond(s, \varepsilon) \wedge \Diamond$ “le chemin $\pi_{(s,\varepsilon)}$ est emprunté”$)$. Ainsi, avec probabilité 1, à force d'essayer de suivre un chemin π_τ menant à A, l'ensemble de configurations A sera atteint. L'adversaire $\mathcal{U}$ décrit ci-dessus est sans mémoire (grâce aux hypothèses sur les chemins π_τ choisis) et satisfait : $\Pr_{\mathcal{U}}(\sigma \models \Diamond A) = 1$.

(1) $\Rightarrow$ (3) : Supposons qu'il existe un adversaire $\mathcal{U}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Diamond A) = 1$, c'est-à-dire que $\sigma \in \mathbf{E}[\Box A]$. On définit $T \stackrel{\text{def}}{=} \{\tau \in \mathsf{Conf} \mid \Pr_{\mathcal{U}}(\sigma \models \neg A \textit{ Until } \tau) > 0\}$, l'ensemble des configurations qu'il est possible de visiter avant d'atteindre A. Alors $\sigma \in T$ (peu importe si $\sigma \in A$). Montrons que $T \subseteq E$. Il suffit pour cela de montrer que, pour chaque configuration $\tau \in T$, il existe un chemin $\pi_\tau : \tau \stackrel{\text{def}}{=} \tau_0 \xrightarrow{\delta_1} \tau_1 \cdots \xrightarrow{\delta_n} \tau_n \in A$ tel que pour tout i, $Post[\delta_{i+1}](\tau_i) \subseteq T$. Soit $\tau \in T$. Puisque la probabilité est non nulle d'atteindre τ avant A, on a $\Pr_{\mathcal{U}'}(\tau \models \Diamond A) = 1$ où $\mathcal{U}'$ est le suffixe de $\mathcal{U}$ à partir de τ : $\mathcal{U}'(\tau \xrightarrow{*} \tau') = \mathcal{U}(\sigma \xrightarrow{*} \tau \xrightarrow{*} \tau')$ pour un chemin fixé de σ à τ. Ainsi, il existe un chemin menant de τ à A : $\tau = \tau_0 \xrightarrow{\delta_1} \tau_1 \cdots \xrightarrow{\delta_n} \tau_n \in A$ et τ_n est la première configuration de A le long de ce chemin. On observe que chaque τ_i est dans T et que $Post[\delta_{i+1}](\tau_i) \subseteq T$. L'ensemble T est donc un post-point fixe de $X \to \widehat{Pre}^*_X(A)$. Puisque E en est le plus grand point fixe, on obtient : $\sigma \in T \subseteq E$. $\square$

Corollaire 5.30. *Soient $\mathcal{N}$ un NPLCS et R une* région. *Alors l'ensemble des configurations $\sigma \in \mathsf{Conf}$ telles qu'il existe un adversaire $\mathcal{U}$ satisfaisant* $\Pr_{\mathcal{U}}(\sigma \models \Diamond R) = 1$ *est une région effectivement calculable.*

Une conséquence des Corollaires 5.19, 5.26, 5.28 et 5.30 est la décidabilité des quatre problèmes de vérification qualitative de propriétés d'accessibilité pour les NPLCS.

Théorème 5.31 (Décidabilité de l'accessibilité). *Le problème de savoir, étant donné un NPLCS $\mathcal{N}$, une configuration σ et une* région $R \subseteq \mathsf{Conf}$, *si, pour les propriétés d'accessibilité (a) à (d), il existe un adversaire $\mathcal{U}$ satisfaisant*

(a) $\exists \mathcal{U}$ *tel que* $\Pr_{\mathcal{U}}(\sigma \models \Diamond A) > 0$

(b) $\exists \mathcal{U}$ *tel que* $\Pr_{\mathcal{U}}(\sigma \models \Diamond A) = 0$

(c) $\exists \mathcal{U}$ *tel que* $\Pr_{\mathcal{U}}(\sigma \models \Diamond A) < 1$

(d) $\exists \mathcal{U}$ *tel que* $\Pr_{\mathcal{U}}(\sigma \models \Diamond A) = 1$

est décidable.

De plus, dans les quatre cas, l'existence d'un adversaire est équivalente à l'existence d'un adversaire sans mémoire pour la même propriété.

Preuve : Pour les quatre variantes qualitatives, les Corollaires 5.19, 5.26, 5.28 et 5.30 établissent la calculabilité de la région constituée des configurations pour lesquelles il existe un adversaire $\mathcal{U}$ satisfaisant. Il suffit alors de tester l'appartenance de σ à la région ainsi calculée pour répondre au problème. □

5.2.2 Conjonction d'accessibilité

La décidabilité s'étend de l'accessibilité à des conjonctions de propriétés d'accessibilité. Autrement dit, si on considère $\phi = \bigwedge_{1\leq i\leq n} \Diamond A_i$, les quatre variantes de vérification qualitative sont décidables. Ici encore, on utilise le Théorème 1.26, concernant le fragment gardé du μ-calcul, pour montrer la convergence des points fixes définissant l'ensemble des configurations satisfaisantes. Cependant, une différence avec l'accessibilité est la nécessité d'utiliser des adversaire plus compliqués, c'est-à-dire à mémoire finie plutôt que sans mémoire.

Théorème 5.32. *Le problème de savoir étant donné un NPLCS $\mathcal{N}$, une configuration σ et des* régions $R_1, \cdots, R_n \subseteq \mathsf{Conf}$*, si, pour les propriétés d'accessibilité généralisée* (a) *à* (d)*, il existe un adversaire $\mathcal{U}$ satisfaisant*

(a) $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i=1}^{n} \Diamond R_i) > 0$*, ou*

(b) $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i=1}^{n} \Diamond R_i) = 0$*, ou*

(c) $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i=1}^{n} \Diamond R_i) < 1$*, ou*

(d) $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i=1}^{n} \Diamond R_i) = 1$.

est décidable.

De plus, l'existence d'un adversaire satisfaisant (a) *(resp.* (b), (c), (d)*) entraîne l'existence d'un adversaire à mémoire finie pour* (a) *(resp.* (b), (c), (d)*).*

Les cas $n = 1$ ont été traités dans la sous-section précédente. Nous prouvons les quatre assertions tour à tour, soit en réduisant le problème au cas $n = 1$ déjà traité, soit en donnant, comme c'était le cas pour l'accessibilité, des termes de L_μ définissant les ensembles de configurations pour lesquelles un adversaire existe.

Soient $A_1, \cdots, A_n \subseteq \mathsf{Conf}$ des ensembles de configurations quelconques. On considère la conjonction d'accessibilité $\Diamond A_1 \wedge \cdots \wedge \Diamond A_n$.

Preuve de (a) : Le cas de plusieurs propriétés $\Diamond A_1, \cdots, \Diamond A_n$ peut être réduit au cas $n = 1$ en construisant le produit de $\mathcal{N}$ avec un automate se souvenant des A_i visités jusqu'alors. La taille de ce nouveau NPLCS $\mathcal{N}'$ est celle de $\mathcal{N}$ multipliée par 2^n, le nombre de sous-ensembles de $\{1, \cdots, n\}$. Il existe un adversaire sans mémoire pour $\mathcal{N}'$, grâce au Lemme 5.29 (cf. page 95). On en déduit un adversaire à mémoire finie (dont les modes sont les sous-ensembles de $\{1, \cdots, n\}$) pour $\mathcal{N}$ satisfaisant $\bigwedge_{1\leq i\leq n} \Diamond A_i$ presque sûrement.

En général, l'existence d'un adversaire à mémoire finie n'implique pas l'existence d'un adversaire sans mémoire. On illustre cette constatation à la Figure 5.5. En effet, si $A = \{s_A\}$

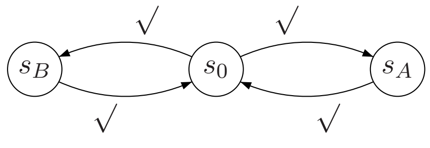

FIG. 5.5 – Les adversaires sans mémoire ne suffisent pas pour $\Diamond A \wedge \Diamond B$ presque sûrement

et $B = \{S_B\}$, pour la propriété $\Diamond A \wedge \Diamond B$, il n'existe pas d'adversaire sans mémoire pour lequel A et B soient visités presque sûrement. La seule possibilité pour vérifier les deux contraintes est de faire des choix différents lors de deux visites de s_0.
Preuve de (b) : On récrit la question en :

$$\exists \mathcal{U} \text{ s.t. } \Pr_{\mathcal{U}}(\sigma \models \bigvee_{1 \leq i \leq n} \Box B_i) = 1 \qquad\qquad \text{où } B_i \stackrel{\text{def}}{=} \overline{A_i}.$$

Pour I, sous-ensemble non vide de $\{1, \cdots, n\}$, on définit de façon inductive (sur la taille de I) les ensembles :

$$X_{\{i\}} \stackrel{\text{def}}{=} \nu X.\widehat{Pre}_X(\mathsf{Conf}) \wedge B_i$$
$$X_I \stackrel{\text{def}}{=} \nu X.\widehat{Pre}_{\cup_{\emptyset \neq J \subseteq I} X_J}(\mathsf{Conf}) \wedge \bigwedge_{i \in I} B_i$$

Lemme 5.33. *Il existe un adversaire $\mathcal{U}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \bigvee_{1 \leq i \leq n} \Box B_i) = 1$ si et seulement s'il existe un sous-ensemble I non vide de $\{1, \cdots n\}$ tel que $\sigma \in X_I$.*

Preuve : Le cas $n = 1$ a été traité précédemment (cf. Lemme 5.23, page 93).
($\Longleftarrow$) : Supposons que $\sigma \in X_I$ pour $I \neq \emptyset$. Décrivons un adversaire $\mathcal{U}$ qui assure $\bigvee_{1 \leq i \leq n} \Box B_i$ presque sûrement. Puisque $\sigma \in X_I$, il existe une règle de transition δ tirable en σ telle que $Post[\delta](\sigma) \subseteq \bigcup_{\emptyset \neq J \subseteq I} X_J$. L'adversaire $\mathcal{U}$ dans la configuration σ choisit cette règle δ et atteint X_J pour un sous-ensemble non vide J de I (J peut dépendre des pertes). Si $J \subsetneq I$, une induction sur la taille de I et le cas de base ($sizeof I = 1$) permettent de conclure. S'il existe une exécution π qui reste pour toujours dans X_I, alors π satisfait $\Box \bigwedge_{i \in I} B_i$.
Dans une configuration $\tau \in X_J$, $\mathcal{U}$ fait donc le choix d'une transition δ (qui peut dépendre de J) qui ne permet que d'atteindre $\bigcup_{\emptyset \neq J' \subseteq J} X_{J'}$. L'adversaire ainsi construit satisfait $\bigcup_{i \in I} \Box B_i$ sûrement, et donc presque sûrement. Il est à mémoire finie ; il y a un mode pour chaque ensemble non vide $J \subseteq \{1, \cdots, n\}$.
($\Longrightarrow$) : Définissons, pour $I \neq \emptyset$, l'ensemble des configurations :

$$T_I \stackrel{\text{def}}{=} \{\tau \in \mathsf{Conf} | \Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i \in I} \Box B_i \wedge \Diamond \tau) > 0\}.$$

Par hypothèse, $\sigma \in T_{I_0}$ pour un certain I_0. Montrons que $T_I \subseteq \widehat{Pre}_{\bigcup_{\emptyset \neq J \subseteq I} T_J}(\mathsf{Conf}) \wedge \bigwedge_{i \in I} B_i$. Ainsi, en tant que post-point fixe, T_I sera un sous-ensemble du plus grand point fixe, c'est-à-dire X_I. Soit donc $\tau \in T_I$, pour $I \neq \emptyset$. Par définition, il existe un chemin conforme à $\mathcal{U}$, $\pi : \ \sigma \xrightarrow{*} \tau$ tel que $\pi \models \bigwedge_{i \in I} \Box B_i$. En particulier, $\tau \in \bigcap_{i \in I} B_i$. De plus, puisque la probabilité est non nulle de prendre ce chemin depuis σ, nécessairement, il existe une règle de transition tirable en τ telle que les successeurs satisfont chacun $\bigwedge_{i \in J} B_i$ pour $J \subseteq I$ non vide. Si $\tau' \in Post[\delta](\tau)$, et π' est le chemin π augmenté de $\tau \to \tau'$, on a $\pi' \models \bigwedge_{i \in J} \Box B_i$, pour $\emptyset \neq J \subseteq I$. Ainsi, $Post[\delta](\tau) \subseteq \bigcup_{\emptyset \neq J \subseteq I} T_J$.
On conclut que pour tout I, $T_I \subseteq X_I$. Et donc $\sigma \in T_{I_0} \subseteq X_{I_0}$. $\Box$

Remarque 5.34. *Comme dans le cas précédent, les adversaires sans mémoire ne sont pas aussi puissants aux adversaires généraux pour le problème de la vérification de $\Diamond A \wedge \Diamond B$ avec probabilité nulle. Cependant, pour donner un contre-exemple, il faut ici choisir des régions* *plus compliquées que les régions de contrôle. On prend pour exemple le NPLCS représenté Figure 5.6, avec les régions*

$$R \stackrel{\text{def}}{=} (s_0, \uparrow\varepsilon) + (s_1, \uparrow m) + (s_2, \uparrow\varepsilon) + (s_3, \uparrow\varepsilon) \quad et,$$
$$R' \stackrel{\text{def}}{=} (s_0, \uparrow\varepsilon) + (s_1, \uparrow\varepsilon - \uparrow m) + (s_2, \uparrow\varepsilon) + (s_4, \uparrow\varepsilon).$$

Pour cet exemple, un adversaire satisfaisant $\Box R \vee \Box R'$ presque sûrement nécessite de la mémoire pour savoir si, lors du passage dans l'état de contrôle s_1, le canal contenait un m ou non.

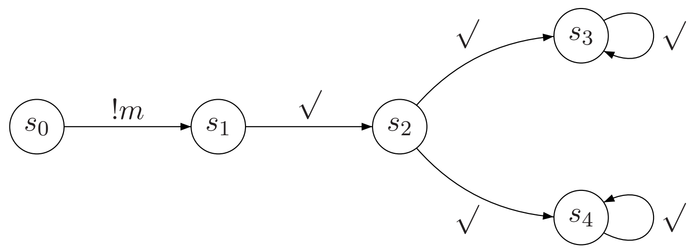

FIG. 5.6 – Les adversaires sans mémoire ne suffisent pas pour $\Box R \vee \Box R'$ presque sûrement

Les ensembles X_I sont définis par des termes gardés de L_μ. Ainsi, si on considère des régions $R'_1, \cdots \mathsf{R}'_n$ (plutôt que des ensembles quelconques de configurations $B_1, \cdots, B_n$), le Théorème 1.26 (cf. page 32) implique la calculabilité des X_I.

Ceci permet de conclure quant à la décidabilité de (b).

Preuve de (c) : Le problème (c) se ramène facilement à un problème déjà traité (invariant avec probabilité positive). En effet,

$$\mathrm{Pr}_{\mathcal{U}}(\sigma \models \bigwedge_{1\leq i\leq n} \Diamond A_i) < 1 \text{ ssi } \exists i \ \mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond A_i) < 1.$$

Le Théorème 5.31 (cf. page 96, propriété (c)) permet de conclure.

Preuve de (d) : Le cas $n = 1$ a déjà été traité. Rappelons qu'il existe un adversaire $\mathcal{U}$ satisfaisant $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond A) = 1$ si et seulement si $\sigma \in \nu Y.\widehat{Pre}^*_Y(A)$. On définit, de façon inductive, des ensembles T_I pour J sous-ensemble non vide de $\{1, \cdots, n\}$ par :

$$\begin{cases} T_{\{i\}} \stackrel{\text{def}}{=} \nu Y.\widehat{Pre}^*_Y(A_i), & \text{et} \\ T_I \stackrel{\text{def}}{=} \nu Y.\widehat{Pre}^*_Y\big(\bigvee_{i\in I}(A_i \wedge T_{I\setminus\{i\}})\big) & \text{si } |I| > 1. \end{cases}$$

On va montrer que les T_I ainsi construits représentent les ensembles de configurations pour lesquelles il existe un adversaire $\mathcal{U}$ satisfaisant $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \bigwedge_{i\in I} \Diamond A_i) = 1$. Ils sont construits inductivement en tenant compte de tous les ordres possibles dans lesquels les A_i seront visités. En effet, étant donnée une configuration initiale σ, pour un adversaire fixé il se peut que cet ordre dépende des pertes, et donc soit différent entre les exécutions.

Lemme 5.35. *Il existe un adversaire $\mathcal{U}$ tel que $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \bigwedge_{i\in I} \Diamond A_i) = 1$ si et seulement si, $\sigma \in T_I$.*

Preuve : La démonstration se fait par récurrence sur la taille de l'ensemble I. Pour un singleton, on utilise le résultat énoncé plus tôt dans le Lemme 5.29 (cf. page 95).

Soit I un sous-ensemble de $\{1, \cdots, n\}$ ayant au moins deux éléments. On commence par montrer que si $\sigma \in T_I$, il existe un adversaire satisfaisant $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i\in I} \Diamond A_i) = 1$. Toujours grâce au Lemme 5.29, on sait qu'il existe un adversaire sans mémoire $\mathcal{V}$ tel que $\Pr_{\mathcal{V}}(\sigma \models \Diamond(\bigvee_{i\in I}(A_i \wedge T_{I\setminus\{i\}})) = 1$. Une fois cet ensemble atteint (ce qui arrive presque sûrement), on applique l'hypothèse de récurrence pour conclure. En effet, les exécutions visitent alors un ensemble A_i (l'indice dépendant de l'exécution) tout en étant dans $T_{I\setminus\{i\}}$ auquel on peut appliquer l'hypothèse de récurrence pour exhiber un adversaire $\mathcal{V}_{I\setminus\{i\}}$. En mettant bout à bout l'adversaire $\mathcal{V}$ et les adversaires $\mathcal{V}_{I\setminus\{i\}}$, on obtient un adversaire vérifiant la bonne propriété.

Réciproquement, supposons qu'il existe un adversaire $\mathcal{U}$ pour lequel $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i\in I} \Diamond A_i) = 1$. On définit l'ensemble de configurations $T \stackrel{\text{def}}{=} \{\tau \in \mathsf{Conf} | \Pr_{\mathcal{U}}(\sigma \models (\bigwedge_{i\in I} \overline{A_i}) \textit{ Until } \tau) > 0\}$, autrement dit l'ensemble des configurations que l'on peut visiter sous l'action de $\mathcal{U}$ avant de visiter les ensembles A_i pour $i \in I$. Tout d'abord, $\sigma \in T$. Montrons que T est un post-point fixe de $Y \to \widehat{Pre}^*_Y\big(\bigvee_{i\in I}(A_i \wedge T_{I\setminus\{i\}})\big)$. À partir de toute configuration $\tau \in T$, $\mathcal{U}$ est capable de visiter chacun des A_i pour $i \in I$ presque sûrement. De plus, par définition de T, toutes les configurations rencontrées avant de visiter un des A_i sont encore dans T. Ainsi $T \subseteq Pre^*_T(\bigvee A_i)$. Mais, lorsque la première configuration de A_i (pour i donné) est visitée, $\mathcal{U}$ doit être capable d'assurer $\bigwedge_{j\in I\setminus\{i\}\Diamond A_j}$ presque sûrement. Ce qui est exactement

$$T \subseteq \widehat{Pre}^*_T\big(\bigvee_{i\in I}(A_i \wedge T_{I\setminus\{i\}})\big).$$

Ainsi, T est un post-point fixe d'une fonctionnelle dont T_I est le plus grand point fixe. On en déduit $T \subseteq T_I$. Finalement $\sigma \in T \subseteq T_I$. $\square$

5.2.3 Accessibilité répétée

Nous avons vu précédemment que le problème de l'accessibilité répétée avec probabilité positive était indécidable (cf. Théorème 5.6, page 84). Nous montrons ici, que les trois autres problèmes (probabilité 1, probabilité < 1 et probabilité 0) sont décidables.

Dans un deuxième temps, on montrera que le problème de l'accessibilité répétée avec probabilité positive est décidable lorsqu'on se restreint à la classe des adversaires à mémoire finie.

Si $A \subseteq \mathsf{Conf}$ est un ensemble de configurations, on note $\mathbf{E}[\Box\Diamond A]$, $T^{<1}_{\Box\Diamond A}$, et $T^{=0}_{\Box\Diamond A}$ les ensembles de configurations σ telles qu'il existe un adversaire $\mathcal{U}$ satisfaisant $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A) = 1, < 1, = 0$, respectivement. Remarquons que $T^{=0}_{\Box\Diamond A}$ peut aussi se noter $\mathbf{E}[\Diamond\Box\overline{A}]$.

Ces trois ensembles s'expriment par des termes du μ-calcul. On le montre tour à tour.

Lemme 5.36. *Soit $A \subseteq \mathsf{Conf}$ un ensemble de configurations. Alors $\mathbf{E}[\Box\Diamond A] = \nu X.\widehat{Pre}^+_X(A)$.*

Preuve : Commençons par la première égalité. Soit $\sigma \in \mathbf{E}[\Box\Diamond A]$. Par définition, il existe un adversaire $\mathcal{U}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A) = 1$. Définissons alors l'ensemble $T \subseteq \mathsf{Conf}$ par

$$T \stackrel{\text{def}}{=} \{\tau | \Pr_{\mathcal{U}}(\sigma \models \Diamond\tau) > 0\}.$$

Nécessairement, T intersecte A non trivialement. De plus, la configuration initiale σ appartient à T. On montre que T est un sous-ensemble de $\nu X.\widehat{Pre}^+_X(A)$. Pour cela, il suffit de montrer que $T \subseteq \widehat{Pre}^+_T(A)$ (*i.e.* qu'il est post-point fixe). Soit donc $\tau \in T$. Le fait que $\Pr_{\mathcal{U}}(\sigma \models \Diamond\tau) > 0$

entraîne $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond\tau \wedge \Box\Diamond A) > 0$, donc il existe un chemin non vide conforme à $\mathcal{U}$ menant de τ à A :

$$\pi : \tau \stackrel{\text{def}}{=} \tau_0 \xrightarrow{\delta_1} \tau_1 \cdots \xrightarrow{\delta_m} \tau_m \in A \qquad \text{avec } m \geq 1.$$

Si alors τ' est une configuration d'un $Post[\delta_i](\tau_{i-1})$ pour $i \geq 1$, c'est-à-dire si τ' peut être obtenu en tirant les règles données par le chemin π, on a

$$\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond\tau') > 0$$

ce qui est exactement $\tau' \in T$. Ainsi $\tau \in \widehat{Pre}^{+}_{T}(A)$. On en conclut que $T \subseteq \nu X.\widehat{Pre}^{+}_{X}(A)$, et comme $\sigma \in T$, on obtient $\sigma \in \nu X.\widehat{Pre}^{+}_{X}(A)$.

Réciproquement, soit σ une configuration de l'ensemble $\nu X.\widehat{Pre}^{+}_{X}(A)$, que l'on note E dans le reste de cette preuve. Nous allons construire un adversaire (sans mémoire) $\mathcal{U}$ qui permet d'avoir $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond A) = 1$. Pour cela, à chaque configuration $\tau \in E$, nous associons un chemin simple $\pi_\tau :\ \tau \xrightarrow{*} A$ qui témoigne de l'appartenance de τ à E. Dans le choix des chemins π_τ, on ajoute la contrainte que si $\tau' \in E$ apparaît le long de π_τ, alors $\pi_{\tau'}$ est le suffixe de π_τ commençant en τ'. Ces chemins décrivent les choix que $\mathcal{U}$ fait : dans la configuration τ, $\mathcal{U}$ tire la première règle de transition du chemin π_τ. Ainsi défini, $\mathcal{U}$ est sans mémoire. Il reste à prouver qu'il donne $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond A) = 1$. Pour cela, si I est un sous-ensemble de S, on définit

$$\Pi_I \stackrel{\text{def}}{=} \{\pi | \mathtt{Inf}(\pi) \cap S_\varepsilon = I\}.$$

Π_I est l'ensemble des exécutions conformes à U qui ont pour configurations vides visitées infiniment souvent exactement I_ε. Les Π_I sont des ensembles disjoints et la propriété de l'attracteur fini entraîne :

$$\mathrm{Pr}_{\mathcal{U}}(\bigcup_{\emptyset \neq I \subseteq S} \Pi_I) = 1.$$

Fixons $I \subseteq S$ non vide, et considérons Π_I. Si $s \in I$, tous les successeurs possibles par $\mathcal{U}$ de (s, ε) seront visités infiniment souvent autant que (s, ε) (cf. Proposition 4.12, page 79). Plus précisément :

$$\begin{aligned}\mathrm{Pr}_{\mathcal{U}}(\Pi_I) &= \mathrm{Pr}_{\mathcal{U}}(\Pi_I \wedge \bigwedge_{s \in I} \bigwedge_{\tau \in S((s,\varepsilon))} \Box\Diamond\tau) \\ &= \mathrm{Pr}_{\mathcal{U}}(\Pi_I \wedge \Box\Diamond A)\end{aligned}$$

puisque parmi les successeurs, il y a en particulier des configurations de A (celles qui ont permis de définir les chemins π_τ). On conclut en « recollant les morceaux » :

$$\begin{aligned}\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond A) &= \mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond A \wedge \bigcup_{\emptyset \neq I \subseteq S} \Pi_I) \\ &= \sum_{\emptyset \neq I \subseteq S} \mathrm{Pr}_{U}(\sigma \models \Box\Diamond A \wedge \Pi_I) \\ &= \sum_{\emptyset \neq I \subseteq S} \mathrm{Pr}_{U}(\Pi_I) \\ &= \mathrm{Pr}_{\mathcal{U}}(\bigcup_{\emptyset \neq I \subseteq S} \Pi_I) \\ &= 1.\end{aligned}$$

□

Corollaire 5.37. *Soit* $R \subseteq \mathsf{Conf}$ *une* région. *Alors* $\mathbf{E}[\Box\Diamond R]$ *est une région effectivement calculable.*

Preuve : Le lemme qui précède donne $\mathbf{E}[\Box\Diamond R] = \nu X.\widehat{Pre}^{+}_{X}(R)$. On peut récrire ce terme en un terme gardé, en utilisant la Proposition 5.22 (cf. page 93)

$$\mathbf{E}[\Box\Diamond R] = \nu X.\widehat{Pre}^{+}_{K_{\downarrow}X}(R).$$

Le Théorème 1.26 (cf. page 32) s'applique donc pour donner le résultat. $\square$

Ce résultat s'étend à une conjonction de propriétés d'accessibilité répétée.

Lemme 5.38. *Soient* $A_1, \cdots, A_n \subseteq \mathsf{Conf}$ *des ensembles de configurations. Alors,*

$$\mathbf{E}[\bigwedge_{i=1}^{n} \Box\Diamond A_i] = \nu X.\widehat{Pre}^{+}_{X}\Big(A_1 \wedge \big(\widehat{Pre}^{+}_{X}(A_2) \wedge (\cdots \widehat{Pre}^{+}_{X}(A_n))\big)\Big).$$

Les techniques utilisées pour la démonstration du Lemme 5.38 sont similaires à celles de la preuve du Lemme 5.36 (cf. page 100).

Corollaire 5.39. *Soient* $R_1, \cdots, R_n \subseteq \mathsf{Conf}$ *des* régions. *Alors l'ensemble* $\mathbf{E}[\bigwedge_{i=1}^{n} \Box\Diamond R_i]$ *est une région effectivement calculable.*

Preuve : Le terme donné dans le Lemme 5.38 pour $\mathbf{E}[\bigwedge_{i=1}^{n} \Box\Diamond R_i]$ peut être gardé en utilisant que $\widehat{Pre}^{+}_{X}(R) = \widehat{Pre}^{+}_{K_{\downarrow}X}(R)$ (cf. Proposition 5.22, page 93). Le Théorème 1.26 s'applique donc pour prouver la calculabilité de $\mathbf{E}[\bigwedge_{i=1}^{n} \Box\Diamond R_i]$. $\square$

Lemme 5.40. *Soit* $A \subseteq \mathsf{Conf}$ *un ensemble de configurations. Alors* $T^{<1}_{\Box\Diamond A} = Pre^*(\mathbf{E}[\Box\overline{A}]) = \mu X.(\mathbf{E}[\Box\overline{A}] \vee Pre(X))$.

Preuve : Si $\mathcal{U}$ est un adversaire et σ une configuration, alors

$$\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond A) < 1 \text{ ssi } \mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond\Box\overline{A}) > 0.$$

On montre facilement que pour $\sigma \in \mathsf{Conf}$, il existe $\mathcal{U}$ tel que $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond\Box\overline{A}) > 0$ si et seulement si $\sigma \in Pre^*(\mathbf{E}[\Box\overline{A}])$, c'est-à-dire si σ peut atteindre une configuration à partir de laquelle il existe un adversaire assurant $\Box\overline{A}$ presque sûrement. $\square$

Corollaire 5.41. *Soit* $R \subseteq \mathsf{Conf}$ *une* région. *Alors* $T^{<1}_{\Box\Diamond R}$ *est une région calculable.*

Preuve : Le lemme qui précède nous permet d'écrire $T^{<1}_{\Box\Diamond R} = Pre^*(\mathbf{E}[\Box\overline{R}])$. Or, d'après le Corollaire 5.26 (cf. page 94), $\mathbf{E}[\Box\overline{R}]$ est calculable et les prédécesseurs itérés également. $\square$

Lemme 5.42. *Soit* $A \subseteq \mathsf{Conf}$ *un ensemble de configurations. Alors* $T^{=0}_{\Box\Diamond A} = \mathbf{E}[\Diamond\mathbf{E}[\Box\overline{A}]]$.

Preuve : Si $\mathcal{U}$ est un adversaire et σ une configuration, alors

$$\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond A) = 0 \text{ ssi } \mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond\Box\overline{A}) = 1.$$

Ensuite, pour $\sigma \in \mathsf{Conf}$, il existe un adversaire tel que $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond\Box\overline{A}) = 1$ si et seulement si, à partir de σ, on atteint presque sûrement une configuration, à partir de laquelle il existe un adversaire assurant $\Box\overline{A}$ presque sûrement. $\square$

Corollaire 5.43. *Soit $R \subseteq \mathsf{Conf}$ une* région. *Alors $T^{=0}_{\Box\Diamond R}$ est une région effectivement calculable.*

Preuve : D'après le lemme précédent, $T^{=0}_{\Box\Diamond A} = \mathbf{E}[\Diamond \mathbf{E}[\Box \overline{A}]]$ qui est calculable grâce aux Corollaires 5.30 (cf. page 96) et 5.26 (cf. page 94). □

Théorème 5.44. *Le problème de savoir, étant donné un NPLCS $\mathcal{N}$, une configuration σ et une* région *R, si pour les propriétés d'accessibilité répétée (a) à (c) il existe un adversaire $\mathcal{U}$ satisfaisant*

(a) $\exists \mathcal{U},\ \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond R) = 1$, *ou*

(b) $\exists \mathcal{U},\ \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond R) < 1$, *ou*

(c) $\exists \mathcal{U},\ \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond R) = 0$.

est décidable.

De plus, l'existence d'un adversaire satisfaisant (a) (resp. (b) ou (c)) entraîne l'existence d'un adversaire à mémoire finie satisfaisant (a) (resp. (b) ou (c)).

Preuve : C'est une conséquence des Corollaires 5.37, 5.41 et 5.43. □

5.2.4 Complexité

Nous nous intéressons à présent à la complexité des algorithmes de décision pour les cas d'accessibilité et accessibilité répétée présentés jusqu'ici.

Pour presque tous les problèmes étudiés[1], nous montrons qu'ils sont non primitif récursifs, comme la plupart des problèmes sur les LCS. Pour cela, nous utilisons le fait que l'accessibilité d'un état de contrôle dans un LCS est un problème non primitif récursif [Sch02]. La réduction utilisera, comme c'était le cas dans les résultats d'indécidabilité, le gadget de nettoyage $\mathsf{Net}_{\mathsf{M}}$. Notons que la borne inférieure de complexité de ces problèmes s'applique toujours lorsqu'on restreint les ensembles de configurations à des ensembles d'états de contrôle.

Théorème 5.45 (Complexité)**.** *Le problème, étant donné un NPLCS $\mathcal{N}$, une configuration $\sigma \in \mathsf{Conf}$ et un ensemble $A \subseteq S$ d'états de contrôle, de savoir s'il existe un adversaire $\mathcal{U}$ satisfaisant* (a.1) *(ou* (a.2) $\cdots$ *ou* (b.3)*) est non primitif récursif.*

(a.1)	$\Pr_{\mathcal{U}}\big(\sigma \models \Diamond A\big) > 0$, *ou*	(b.1)	$\Pr_{\mathcal{U}}\big(\sigma \models \Box\Diamond A\big) = 0$, *ou*
(a.2)	$\Pr_{\mathcal{U}}\big(\sigma \models \Diamond A\big) = 1$, *ou*	(b.2)	$\Pr_{\mathcal{U}}\big(\sigma \models \Box\Diamond A\big) = 1$, *ou*
(a.3)	$\Pr_{\mathcal{U}}\big(\sigma \models \Diamond A\big) < 1$, *ou*	(b.3)	$\Pr_{\mathcal{U}}\big(\sigma \models \Box\Diamond A\big) < 1$.

Dans tous les cas, on réduit le problème de l'accessibilité d'un état de contrôle dans les LCS, qui est connu pour être non primitif récursif [Sch02].

Le cas $(a.1)$ est le plus simple. En effet par le Lemme 5.18 (cf. page 92), le problème de l'existence d'un adversaire permettant d'atteindre A avec probabilité positive est équivalent à l'accessibilité de A à partir de σ dans le LCS sous-jacent à $\mathcal{N}$.

Pour le cas $(a.3)$, nous utilisons la réduction décrite Figure. 5.7.

Soit $\mathcal{L}$ un LCS quelconque. On construit alors $\mathcal{L}'$ à partir de $\mathcal{L}$ de la façon suivante. Tout d'abord, on ajoute trois états de contrôle `puits`, `succes` et `accept`. L'état `accept` est accessible par une transition $\surd$ depuis n'importe quel état de contrôle. Pour ce qui est de

[1] voir la Remarque 5.48 page 106

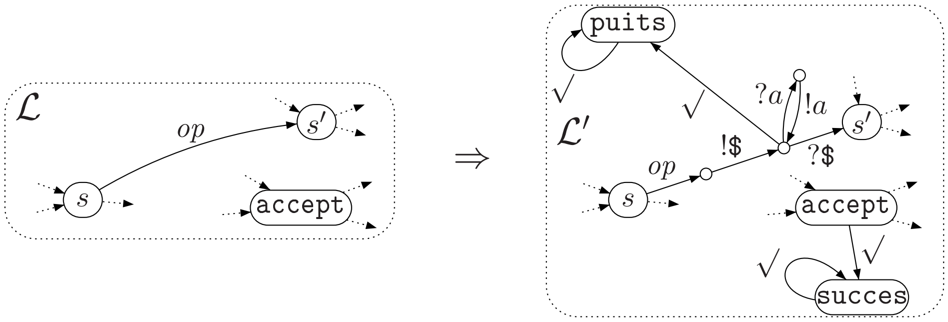

FIG. 5.7 – Construction de $\mathcal{L}'$ à partir d'un LCS $\mathcal{L}$ arbitraire pour le cas $(a.3)$

l'état puits nommé `succes`, il ne peut être atteint qu'à partir de `accept`. Enfin, pour toute transition $\delta : s \xrightarrow{op} s'$ de $\mathcal{L}$, on introduit trois états intermédiaires, l_δ, l'_δ et l''_δ. L'opération op est réalisée entre s et l_δ : $s \xrightarrow{op} l_\delta$. De l_δ à l'_δ on écrit un `$`, qui pourra être lu en passant de l'_δ à s'. Pour cela, on utilise un circuit entre l'_δ et l''_δ qui permet de faire une permutation circulaire des messages dans le canal, de façon à faire apparaître `$` en tête. Il se peut que des messages soient perdus, on peut alors rejoindre l'état de contrôle `puits` pour éviter le blocage.

Le but de cette réduction et d'imposer que `accept` et `succes` sont les seuls états qui peuvent atteindre de façon sûre `succes`. Depuis tous les autres états, les pertes peuvent forcer le système à aller dans l'état puits. Soit s_0 un état de contrôle de $\mathcal{L}$. Le lemme qui suit exprime l'équivalence qui permet de conclure :

Lemme 5.46. *Dans $\mathcal{N} = (\mathcal{L}', \lambda)$ les assertions suivantes sont équivalentes :*

1. $\exists \mathcal{U}\, \Pr_\mathcal{U}((s_0, \varepsilon) \models \Diamond \texttt{puits}) < 1$
2. $s_0 \in \exists\ (\neg \texttt{puits})\ \mathit{Until}\ \{\texttt{accept}, \texttt{succes}\}$
3. $s_0 \xrightarrow{*}_\mathcal{L} \{\texttt{accept}\}$

où $s_0 \xrightarrow{*}_\mathcal{L} \{\texttt{accept}\}$ signifie qu'il existe un chemin menant de s_0 à `accept` *qui ne visite que des états de $\mathcal{L}$.*

Preuve : Pour montrer cette équivalence, on utilise la caractérisation du Lemme 5.27 (cf. page 94). En effet, il existe $\mathcal{U}$ tel que $\Pr_\mathcal{U}((s_0, \varepsilon) \models \Diamond \texttt{puits}) < 1$ si et seulement si il existe $\mathcal{U}$ tel que $\Pr_\mathcal{U}((s_0, \varepsilon) \models \Box \neg \texttt{puits}) > 0$. L'assertion 1. est donc équivalente, grâce au Lemme 5.27 à

$$(s_0, \varepsilon) \in \mu X.\mathbf{E}[\Box \neg \texttt{puits}] \vee (\mathit{Pre}(X) \wedge \neg \texttt{puits}).$$

Or $\mathbf{E}[\Box \neg \texttt{puits}] = \{\texttt{accept}, \texttt{succes}\}$ puisque ce sont les seules configurations de $\mathcal{N}'$ à partir desquelles on peut éviter `puits` presque sûrement. Pour toutes les autres, on court un risque de perdre un message $ et d'être forcé à aller dans `puits`.

On obtient donc l'équivalence de l'assertion 1. avec $(s_0, \varepsilon) \in \mu X.\{\texttt{accept}, \texttt{succes}\} \vee (\mathit{Pre}(X) \wedge \neg \texttt{puits})$ qui est une autre formulation de l'assertion 2.. Ainsi, les assertions 1. et 2. sont équivalentes.

Il reste à observer que 2. et 3. sont équivalentes par construction de $\mathcal{L}'$ à partir de $\mathcal{L}$. En effet, le seul moyen d'atteindre $\{\texttt{accept}, \texttt{succes}\}$ à partir d'une configuration de $\mathcal{L}$ est de passer par `accept`. $\square$

Le problème non primitif récursif « $(s_0, \varepsilon) \xrightarrow{*} (\texttt{accept}, \mathsf{M}^*)$? » se réduit donc à une instance particulière du problème $(a.3)$ dans le Théorème 5.45.

Pour les cas restants, on utilise une autre réduction, qui fait intervenir le gadget de nettoyage Net_M. Soit $\mathcal{L}$ un LCS quelconque possédant un unique canal, et ayant deux états distingués s_0 et `accept`. On construit à partir de $\mathcal{L}$ un nouveau LCS $\mathcal{L}'$ et le NPLCS associé $\mathcal{N}' = (\mathcal{L}', \lambda)$ pour n'importe quel taux de pertes $\lambda \in]0, 1[$. On montre alors que la problème de l'accessibilité d'état de contrôle dans $\mathcal{L}$ (c'est-à-dire `accept` est-il accessible depuis (s_0, ε) ?) est équivalent à des instance particulières des problèmes de vérification quantitative dans $\mathcal{N}'$.

La construction de $\mathcal{L}'$ à partir de $\mathcal{L}$ est représentée Figure. 5.8. $\mathcal{L}'$ utilise le gadget de

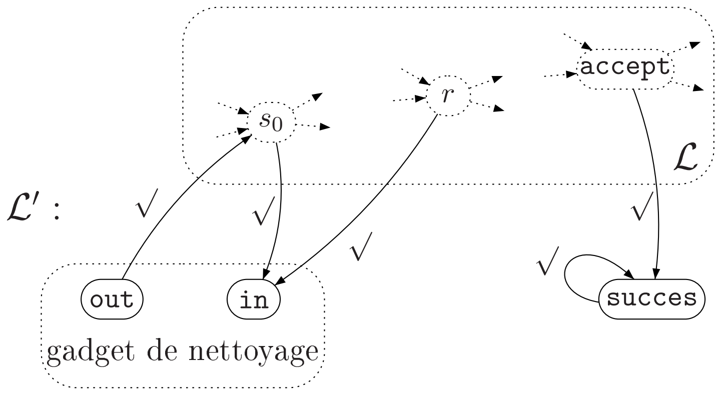

FIG. 5.8 – Le LCS $\mathcal{L}'$ associé à $\mathcal{L}$ dans le Lemme 5.47

nettoyage et un état de contrôle supplémentaire `succes`. À partir de tout état original de $\mathcal{L}$ sauf l'état distingué `accept`, il existe une transition dans $\mathcal{L}'$ menant à l'état initial `in` du gadget de nettoyage. On ajoute également une transition de `out` à s_0. Enfin, depuis `accept`, il existe une transition vers `succes` qui est un état puits.

L'idée de cette réduction est que si `accept` est accessible depuis (s_0, ε) dans $\mathcal{L}$ via un chemin π, un adversaire pourra tenter de réaliser π, et si les pertes ne lui sont pas favorables, tenter à nouveau, autant de fois que nécessaire et passant par le gadget de nettoyage pour revenir à s_0. Les propriétés de Net_M certifient que le retour à s_0 se fait avec le canal vide. D'autre part, par construction de $\mathcal{L}'$, le seul moyen d'atteindre `succes` est de passer par `accept` auparavant. On formalise ces idées dans le lemme suivant :

Lemme 5.47. *Dans le LCS $\mathcal{L}'$ les assertions suivantes sont équivalentes :*

1. $(s_0, \varepsilon) \in Pre^*(\mathbf{E}[\Diamond\{\texttt{succes}\}])$
2. $(s_0, \varepsilon) \in \mathbf{E}[\Diamond\{\texttt{succes}\}]$
3. $(s_0, \varepsilon) \xrightarrow{*} \texttt{succes}$
4. $(s_0, \varepsilon) \xrightarrow{*} \texttt{accept}$
5. $(s_0, \varepsilon) \xrightarrow{*}_{\mathcal{L}} \texttt{accept}$
6. $(s_0, \varepsilon) \in \mathbf{E}[\Box\mathbf{E}[\Diamond\{\texttt{succes}\}]]$
7. $(s_0, \varepsilon) \in Pre^*(\mathbf{E}[\Box\mathbf{E}[\Diamond\{\texttt{succes}\}]])$.

Preuve : (1) $\Rightarrow$ (2) Supposons que $(s_0, \varepsilon) \in Pre^*(\mathbf{E}[\Diamond\{\texttt{succes}\}])$ et soit $(s_0, \varepsilon) \to (q_1, w_1) \to \cdots \to (q_m, w_m)$ avec $(q_m, w_m) \in \mathbf{E}[\Diamond\{\texttt{succes}\}]$ un chemin simple attestant l'accessibilité.

Depuis chaque q_i le long de ce chemin (sauf $\texttt{succes}$), il est possible de retourner à (s_0, ε) en passant par le gadget de nettoyage. On peut donc construire un adversaire qui cherche à atteindre $\mathbf{E}[\Diamond\{\texttt{succes}\}]$ via π en repartant de (s_0, ε) dès que les pertes ne sont pas celles escomptées. Presque sûrement cet adversaire finira par atteindre $\mathbf{E}[\Diamond\{\texttt{succes}\}]$, puis imitera le comportement de l'adversaire qui réalise $\Diamond\{\texttt{succes}\}$ avec probabilité 1. Ainsi, $(s_0, \varepsilon) \in \mathbf{E}[\Diamond\mathbf{E}[\Diamond\{\texttt{succes}\}]]$, et donc $(s_0, \varepsilon) \in \mathbf{E}[\Diamond\{\texttt{succes}\}]$.

(2) $\Rightarrow$ (3) repose sur la définition de $\mathbf{E}[\Diamond\{\texttt{succes}\}]$.

(3) $\Rightarrow$ (4) est évident car $\texttt{succes}$ est uniquement accessible via $\texttt{accept}$.

(4) $\Rightarrow$ (5) Soit π un chemin de (s_0, ε) à $\texttt{accept}$. Si ce chemin sort de $\mathcal{L}$, il passe forcément par le gadget de nettoyage et donc visite à nouveau (s_0, ε). En supposant π simple, il est certain que π reste dans $\mathcal{L}$.

(5) $\Rightarrow$ (6) Supposons que $(s_0, \varepsilon) \xrightarrow{*}_{\mathcal{L}} \texttt{accept}$. Nécessairement, $(s_0, \varepsilon) \xrightarrow{*} \texttt{succes}$ puisque $\texttt{accept} \xrightarrow{\checkmark} \texttt{succes}$. Alors, pour toute configuration σ de $\mathcal{L}'$, σ peut atteindre $\texttt{succes}$. En effet, soit $\sigma = (\texttt{succes}, w)$ ou $\sigma = (\texttt{accept}, w)$ pour un certain w et c'est trivialement vrai, soit σ peut atteindre (s_0, ε) en passant par le gadget de nettoyage puis atteindre $\texttt{succes}$ en utilisant le chemin menant de (s_0, ε) à $\texttt{succes}$. Ainsi, $\mathsf{Conf}_{\mathcal{L}'} \subseteq \mathbf{E}[\Diamond\{\texttt{succes}\}]$ et donc $\mathsf{Conf}_{\mathcal{L}'} \subseteq \mathbf{E}[\Box\mathbf{E}[\Diamond\{\texttt{succes}\}]]$.

(6) $\Rightarrow$ (7) est trivial.

(7) $\Rightarrow$ (1) vient du fait que $A \subseteq \mathbf{E}[\Box A]$ pour tout ensemble $A \subseteq \mathsf{Conf}$ de configurations. $\square$

Montrons à présent comment utiliser ce lemme pour réduire le problème $(s_0, \varepsilon) \xrightarrow{*} \texttt{accept}$ dans $\mathcal{L}$ à des instances de $(a.1)$, $(b.2)$, $(b.1)$ et $(b.3)$ dans L'.

$$\begin{aligned}
\exists\mathcal{U}\ \Pr_{\mathcal{U}}\big((s_0, \varepsilon) \models \Diamond\texttt{succes}\big) = 1& \qquad\qquad \text{(a.2)}\\
\text{ssi } (s_0, \varepsilon) \in \mathbf{E}[\Diamond\{\texttt{succes}\}] &\quad \text{(cf. Lemme 5.29, page 95)}\\
\text{ssi dans } \mathcal{L},\ s_0 \xrightarrow{*} \texttt{accept}.&
\end{aligned}$$

$$\begin{aligned}
\exists\mathcal{U}\ \Pr_{\mathcal{U}}\big((s_0, \varepsilon) \models \Box\Diamond\texttt{succes}\big) = 1& \qquad\qquad \text{(b.2)}\\
\text{ssi } (s_0, \varepsilon) \in \mathbf{E}[\Box\mathbf{E}[\Diamond\{\texttt{succes}\}]] &\quad \text{(cf. Lemme 5.36, page 100)}\\
\text{ssi dans } \mathcal{L},\ s_0 \xrightarrow{*} \texttt{accept}.&
\end{aligned}$$

$$\begin{aligned}
\exists\mathcal{U}\ \Pr_{\mathcal{U}}\big((s_0, \varepsilon) \models \Box\Diamond\neg\texttt{succes}\big) = 0& \qquad\qquad \text{(b.1)}\\
\text{ssi } (s_0, \varepsilon) \in \mathbf{E}[\Diamond\mathbf{E}[\Box(Q \setminus \{\texttt{succes}\})]] &\quad \text{(cf. Lemme 5.42, page 102)}\\
\text{ssi dans } \mathcal{L},\ s_0 \xrightarrow{*} \texttt{accept}.&
\end{aligned}$$

$$\begin{aligned}
\exists\mathcal{U}\ \Pr_{\mathcal{U}}\big((s_0, \varepsilon) \models \Box\Diamond\neg\texttt{succes}\big) < 1& \qquad\qquad \text{(b.3)}\\
\text{ssi } s_0 \xrightarrow{*} Q \setminus \{\texttt{succes}\} &\quad \text{(cf. Lemme 5.40, page 102)}\\
\text{ssi dans } \mathcal{L},\ s_0 \xrightarrow{*} \texttt{accept}.&
\end{aligned}$$

On en déduit que les problèmes $(a.2)$, $(b.2)$, $(b.1)$ et $(b.3)$ sont non primitifs récursifs.

Remarque 5.48. *Dans cette étude de complexité, nous avons considéré tous les problèmes décidables traités jusqu'à présent excepté celui de l'invariant avec probabilité* 1. *Les preuves de borne inférieure de complexité que nous avons présentées sont faites dans le cas où les* *régions sont des régions de contrôle et où la configuration initiale est une configuration de S_0 (ensemble des configurations avec canaux vides). En se plaçant dans ce cadre, on a montré dans [BBS07] (voir le Théorème 5.3) que le problème de la vérification d'invariant presque sûrement était* NLOGSPACE*-complet, puisqu'il est équivalent à une question d'accessibilité dans un sous-graphe du graphe de contrôle du NPLCS. D'ailleurs, pour ce cas particulier, les adversaire aveugles étaient suffisants. Ce n'est pas le cas en général, c'est-à-dire pour des régions régulières quelconques et pour une configuration initiale arbitraire.*

5.3 Vérification qualitative pour des adversaires à mémoire finie

On remarque d'une part que dans les cas décidables vus jusqu'à présent, les adversaires à mémoire finie (et bien souvent même sans mémoire) suffisent. Plus précisément l'existence d'un adversaire est équivalente à l'existence d'un adversaire à mémoire finie ayant la même propriété. D'autre part les adversaires construits dans les preuves d'indécidabilité (cf. Preuves du Lemme 5.8, page 86 et de la Proposition 5.14, page 88) utilisent de façon intensive une mémoire infinie. En effet, ils mémorisent à chaque phase un chemin de longueur de plus en plus grande dans le LCS originel $\mathcal{L}$. Ce type d'adversaire semble très puissant et colle peu à la réalité d'un environnement.

Pour toutes ces raisons, nous faisons le choix dans cette section de restreindre l'étude à la classe des adversaires à mémoire finie. Dans un premier temps, nous revenons sur les premiers problèmes indécidables rencontrés (accessibilité récurrente presque sûrement, par exemple) pour montrer leur décidabilité dans la classe des adversaires à mémoire finie. Ensuite nous étendons les résultats de décidabilité aux propriétés ω-régulières en général.

5.3.1 Retour sur les problèmes indécidables

Théorème 5.49. *Le problème de savoir étant donné un NPLCS $\mathcal{N}$, une configuration $\sigma \in$ Conf et des* régions *$R_1, \cdots, R_n \subseteq \mathsf{Conf}$, de savoir s'il existe un adversaire $\mathcal{U}$* à mémoire finie *tel que $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{1\leq i\leq n} \Box\Diamond R_i) > 0$ est décidable.*

Preuve : On commence par montrer que les assertions suivantes sont équivalentes pour $A_1, \cdots, A_n \subseteq \mathsf{Conf}$ des ensembles de configurations quelconques.

1. Il existe un adversaire sans mémoire $\mathcal{U}$ vérifiant $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{1\leq i\leq n} \Box\Diamond A_i) > 0$,
2. il existe une configuration τ telle que
   (a) $\sigma \xrightarrow{*} \tau$
   (b) il existe un adversaire sans mémoire $\mathcal{V}$ tel que $\Pr_{\mathcal{V}}(\tau \models \bigwedge_{1\leq i\leq n} \Box\Diamond A_i) = 1$.

Cette équivalence permet de prouver le Théorème 5.49. En effet, si $A_1, A_2, \cdots$ sont des régions, on peut, dans un premier temps et grâce au Corollaire 5.39 (cf. page 102) calculer la région $\mathbf{E}[\bigwedge \Box\Diamond A_i]$. On peut ensuite calculer l'ensemble des prédécesseurs de $\mathbf{E}[\bigwedge \Box\Diamond A_i]$, qui constituent encore une région (cf. Corollaire 1.30, page 34). Il suffit alors de tester l'appartenance de σ à cette région. Montrons donc l'équivalence de (1) et (2).

(1) $\Longrightarrow$ (2) : Soit $\mathcal{U}$ un adversaire à mémoire finie vérifiant (1). La propriété de l'attracteur fini et la Proposition 4.12 (cf. page 79) entraînent l'existence d'une état de contrôle s et d'un mode u de $\mathcal{U}$ tels que :

$$\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{1\leq i\leq n} \Box\Diamond A_i \wedge \bigwedge_{t\in T} \Box\Diamond t) > 0$$

où T est l'ensemble des configurations accessibles à partir de $(s,\varepsilon)_u$ par des exécutions conformes à $\mathcal{U}$. Par définition de T, pour tout indice $1 \leq i \leq n$, $T \cap A_i \neq \emptyset$. L'adversaire $\mathcal{U}$, partant de la configuration (s,ε) en mode u visite presque sûrement infiniment souvent chaque configuration de T. On en déduit que les ensembles A_i sont visités infiniment souvent avec probabilité 1. C'est à dire :

$$\Pr_{\mathcal{U}}((s,\varepsilon) \models \bigwedge_{1\leq i\leq n} \Box\Diamond A_i) = 1$$

et on obtient (2).

(2) $\Longrightarrow$ (1) : Soient σ, τ et $\mathcal{V}$ comme dans (2). On définit $\mathcal{U}$ comme l'adversaire qui cherche à suivre le chemin $\pi : \sigma \xrightarrow{*} \tau$ (et y parvient avec probabilité positive), puis se comporte comme $\mathcal{V}$ à partir de τ. Il est possible de plus de choisir $\mathcal{U}$ à mémoire finie puisque $\mathcal{V}$ est sans mémoire, et π ne nécessite qu'un nombre fini d'informations. Clairement, $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{1\leq i\leq n} \Box\Diamond A_i) > 0$. $\square$

On présente maintenant des algorithmes de décision pour les quatre variantes du model-checking qualitatif de propriétés de Streett dans le cas des adversaires à mémoire finie.

Théorème 5.50. *Pour les propriétés qualitatives* (a) *à* (d)*, le problème de savoir, étant donné un NPLCS $\mathcal{N}$, une configuration σ et des* régions $R_1, R'_1, \cdots, R_n, R'_n \subseteq \mathsf{Conf}$*, s'il existe un adversaire à mémoire finie satisfaisant*

(a) $\Pr_{\mathcal{U}}\big(\sigma \models \bigwedge\limits_{1\leq i\leq n} (\Box\Diamond R_i \Rightarrow \Box\Diamond R'_i)\big) < 1$,

(b) $\Pr_{\mathcal{U}}\big(\sigma \models \bigwedge\limits_{1\leq i\leq n} (\Box\Diamond R_i \Rightarrow \Box\Diamond R'_i)\big) = 1$,

(c) $\Pr_{\mathcal{U}}\big(\sigma \models \bigwedge\limits_{1\leq i\leq n} (\Box\Diamond R_i \Rightarrow \Box\Diamond R'_i)\big) > 0$,

(d) $\Pr_{\mathcal{U}}\big(\sigma \models \bigwedge\limits_{1\leq i\leq n} (\Box\Diamond R_i \Rightarrow \Box\Diamond R'_i)\big) = 0$,

est décidable.

Preuve : Nous montrons les assertions du Théorème 5.50 tour à tour.

Preuve de (a) Pour la première assertion, il suffit de considérer le problème suivant : étant donnés $R, R' \subseteq \mathsf{Conf}$ régions, $\sigma \in \mathsf{Conf}$, a-t-on

$$\exists\mathcal{U},\ \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond R \wedge \Diamond\Box\overline{R'}) > 0\ ?$$

En effet, si $\mathcal{U}$ est un adversaire alors $\Pr_{\mathcal{U}}\big(\sigma \models \bigwedge\limits_{1\leq i\leq n} (\Box\Diamond R_i \Rightarrow \Box\Diamond R'_i)\big) < 1$ est équivalent à l'existence d'un indice i tel que $\Pr_{\mathcal{U}}\big(\sigma \models \Box\Diamond R_i \Rightarrow \Box\Diamond R'_i\big) < 1$ ou encore (par dualité) $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond R_i \wedge \Diamond\Box\overline{R'_i})$.

On montre donc que cette dernière question est décidable, ce qui est suffisant pour montrer la décidabilité de (a) dans le Théorème 5.50. Pour cela, on commence par montrer comme souvent, que l'existence d'un adversaire satisfaisant avec probabilité positive une propriété à partir de σ est équivalente à l'existence d'une configurations τ accessible depuis σ, et d'un adversaire qui à partir de τ vérifie la même propriété, presque sûrement.

Lemme 5.51. *Soient $A, B \subseteq \mathsf{Conf}$ quelconques. Les assertions suivantes sont équivalentes :*

1. $\exists\mathcal{U},\ \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A \wedge \Diamond\Box\overline{B}) > 0$,
2. *il existe $\tau \in \mathsf{Conf}$ telle que*

(a) $\sigma \xrightarrow{*} \tau$,

(b) $\exists \mathcal{V},\ \Pr_{\mathcal{U}}(\tau \models \Box\Diamond A \wedge \Box\overline{B}) = 1$.

Preuve : (2) $\Longrightarrow$ (1) Dans ce sens, il suffit de construire $\mathcal{U}$ à partir d'une part de $\mathcal{V}$ et d'autre part d'un chemin simple (*i.e.*, ne visitant les configurations qu'au plus une fois) π menant de σ à τ. Partant de σ, $\mathcal{U}$ réalise, avec probabilité positive le chemin π puis copie le comportement de $\mathcal{V}$ à partir de τ, en oubliant le préfixe π. Clairement $\mathcal{U}$ satisfait $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A \wedge \Diamond\Box\overline{B}) > 0$.

(1) $\Longrightarrow$ (2) On définit l'ensemble de configurations T suivant :

$$T \stackrel{\text{def}}{=} \{\sigma' | \Pr_{\mathcal{U}}(\sigma \models \Box\Diamond\sigma' \wedge \Box\Diamond A \wedge \Diamond\Box\overline{B}) > 0\}.$$

La propriété de l'attracteur fini entraîne la non-vacuité de T. De plus, pour toute configuration $\sigma' \in T$, σ' est accessible à partir de σ. Montrons à présent que si $\sigma' \in T$ alors il existe u mode de $\mathcal{U}$ tel que $\Pr_{\mathcal{U}}(\sigma' \models \Box\Diamond A \wedge \Diamond\Box\overline{B}) = 1$.

Puisque $\mathcal{U}$ est à mémoire finie, il existe un mode u de $\mathcal{U}$ tel que :

$$\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond\sigma'_u \wedge \Box\Diamond A \wedge \Diamond\Box\overline{B}) > 0.$$

En utilisant les propriétés des adversaires à mémoire finie, on en déduit que tous les successeurs (en une ou plusieurs étapes) de σ'_u par $\mathcal{U}$ seront visités infiniment souvent presque sûrement si σ'_u est lui même visité infiniment souvent. Soit, en notant $S(\sigma'_u)$ les successeurs :

$$\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond\sigma'_u \wedge \bigwedge_{\sigma'' \in S(\sigma'_u)} \Box\Diamond\sigma'' \wedge \Box\Diamond A \wedge \Diamond\Box\overline{B}) > 0.$$

Mais alors, tous les successeurs de σ'_u, et σ'_u lui-même sont des configurations de $\overline{B}$, et leur intersection avec A est non vide. En effet, à partir de σ'_u toutes les configurations visitées sont membres de $S(\sigma'_u)$. Ainsi :

$$\Pr_{\mathcal{U}}(\sigma'_u \models \Box\Diamond A \wedge \Box\overline{B}) = 1.$$

On conclut en considérant l'adversaire $\mathcal{V}$ à mémoire finie qui, partant de σ', imite le comportement de $\mathcal{U}$ dans la même configuration en mode u. $\Box$

Il reste maintenant à prouver la décidabilité du problème énoncé ci-dessous :

Étant donnés une configuration σ, des régiosn $R, R' \subseteq \mathsf{Conf}$, existe-t-il un adversaire à mémoire finie $\mathcal{U}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond R \wedge \Box\overline{R'}) = 1$?

Soient $A, B \subseteq \mathsf{Conf}$ des ensembles quelconques de configurations. Définissons $\mathbf{E}[\Box\Diamond A \wedge \Box\overline{B}]$ comme l'ensemble des configurations σ à partir desquelles il existe un adversaire à mémoire finie $\mathcal{U}$ satisfaisant $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A \wedge \Box B) = 1$. On montre que $\mathbf{E}[\Box\Diamond A \wedge \Box\overline{B}]$ est définissable dans L_μ par le terme $\nu X.\overline{B} \wedge \widehat{Pre}^{+}_{X}(A)$.

Tout d'abord soit Y un point fixe de la fonctionnelle $X \to \overline{B} \wedge \widehat{Pre}^{+}_{X}(A)$. On souhaite construire un adversaire (à mémoire finie) pour $\sigma \in Y$ fixé qui satisfasse la propriété donnée. Depuis toute configuration $\sigma \in Y$, l'adversaire $\mathcal{U}$ cherche à atteindre A en restant toujours dans Y. Ceci est possible par définition de Y : à chaque $\sigma \in Y$, on associe un chemin simple π_σ, témoin de l'appartenance de σ à $\overline{B} \wedge \widehat{Pre}^{+}_{Y}(A)$. En implémentant ce comportement, et grâce à la propriété de l'attracteur fini, on obtient un adversaire (à mémoire finie si les chemins ont été choisis avec soin) qui vérifie :

$$\Pr_{\mathcal{U}}(\sigma \models \Box Y \wedge \Box\Diamond Y_\varepsilon) = 1$$

où Y_ε dénote le sous-ensemble des configurations de Y qui ont les canaux vides. Pour un ensemble de chemins visitant un certain (τ, ε) infiniment souvent, presque sûrement le chemin $\pi_{(\tau,\varepsilon)}$ sera réalisé infiniment souvent, et donc A visité infiniment souvent. On obtient donc :

$$\Pr_{\mathcal{U}}(\sigma \models \Box Y \wedge \Box\Diamond A) = 1.$$

L'observation que $Y \subseteq \overline{B}$ suffit à conclure que $Y \subseteq \mathbf{E}[\Box\Diamond A \wedge \Box\overline{B}]$.

Réciproquement, soit $\sigma \in \mathbf{E}[\Box\Diamond A \wedge \Box\overline{B}]$. Il existe donc $\mathcal{U}$, adversaire à mémoire finie tel que $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A \wedge \Box\overline{B}) = 1$. Définissons $T \subseteq \mathsf{Conf}$ par

$$T \stackrel{\text{def}}{=} \{\tau | \Pr_{\mathcal{U}}(\sigma \models \Diamond\tau) > 0\}.$$

Clairement, $\sigma \in T$ et T est un sous-ensemble de $\overline{B}$. De plus, si τ est une configuration de T, il doit exister un chemin $\pi_\tau : \tau = \tau_0 \xrightarrow{\delta_1} \tau_1 \cdots \xrightarrow{\delta_n} \tau_n \in A$ menant de τ à A. Puisque A est visité presque sûrement infiniment souvent, on peut supposer $n \geq 1$, c'est-à-dire que le chemin π_τ n'est pas trivial. De plus, par définition de T, pour tout indice $i \in \{1, \cdots n\}$, $Post[\delta_i](\tau_{i-1}) \subseteq T$. Ainsi $T \subseteq \overline{B} \wedge \widehat{Pre}_T^+(A)$. L'ensemble $\nu X.\overline{B} \wedge \widehat{Pre}_X^+(A)$ est un plus grand point fixe, donc $T \subseteq \nu X.\overline{B} \wedge \widehat{Pre}_X^+(A)$. D'où $\sigma \in \mathbf{E}[\Box\Diamond A \wedge \Box\overline{B}]$ implique $\sigma \in \nu X.\overline{B} \wedge \widehat{Pre}_X^+(A)$.

Finalement, on obtient la caractérisation :

Lemme 5.52. $\mathbf{E}[\Box\Diamond A \wedge \Box\overline{B}] = \nu X.\overline{B} \wedge \widehat{Pre}_X^+(A) = \nu X.\overline{B} \wedge \widehat{Pre}_{K_\downarrow X}^+(A)$.

Puisque le terme $\nu X.\overline{B} \wedge \widehat{Pre}_{K_\downarrow X}^+(A)$ est un terme gardé de L_μ, l'ensemble $\mathbf{E}[\Box\Diamond R \wedge \Box\overline{R'}]$ est une région calculable dès lors que R et R' sont des régions.

Preuve de (b) Soient $A_1, B_1, \cdots, A_n, B_n \subseteq \mathsf{Conf}$ des ensembles de configurations. Pour montrer la décidabilité de (b) (dans le cas où ce sont des régions), on considère, pour tout sous-ensemble $I \subseteq \{1, \cdots, n\}$, l'ensemble de configurations C_I défini par :

$$C_I \stackrel{\text{def}}{=} \{\tau | \exists\mathcal{V} \text{ à mémoire finie tel que } \Pr_{\mathcal{V}}(\tau \models \bigwedge_{i\in I} \Box\Diamond B_i \wedge \bigwedge_{i\notin I} \Diamond\Box\overline{A_i}) = 1\}.$$

Autrement dit, on partitionne $\mathbf{E}[\bigwedge_{1\leq i\leq n} \Box\Diamond A_i \Rightarrow \Box\Diamond B_i]$ en fonction des A_i's qui sont ou non vérifiés infiniment souvent.

Puis on définit C'_I qui diffère de C_I car il ne considère que le comportement au long terme, une fois que les A_i (pour $i \notin I$) sont toujours vrais.

$$C'_I \stackrel{\text{def}}{=} \{\tau | \exists\mathcal{V} \text{ à mémoire finie tel que } \Pr_{\mathcal{V}}(\tau \models \bigwedge_{i\in I} \Box\Diamond B_i \wedge \bigwedge_{i\notin I} \Box\overline{A_i}) = 1\}.$$

On commence par caractériser les C_I et des C'_I par des termes de L_μ.

Lemme 5.53. *Soit $I \subseteq \{1, \cdots, n\}$. Alors,*

- $C'_I = \nu X. \bigwedge_{i\notin I} \overline{A_i} \wedge \bigwedge_{i\in I} \widehat{Pre}_X^+(B_i)$, *et*
- $C_I = \nu Y.\widehat{Pre}_Y^*(C'_I)$.

Preuve : Soit Y un ensemble de configurations satisfaisant $Y = \bigwedge_{i\notin I} \overline{A_i} \wedge \bigwedge_{i\in I} \widehat{Pre}_Y^+(B_i)$. Y est donc un point fixe, pas forcément le plus grand de $X \to \bigwedge_{i\notin I} \overline{A_i} \wedge \bigwedge_{i\in I} \widehat{Pre}_X^+(B_i)$. Montrons qu'on peut construire un adversaire sans mémoire qui permet à partir d'une configuration de Y, de vérifier la propriété avec probabilité 1, c'est-à-dire que $Y \subseteq C'_I$. L'adversaire $\mathcal{U}$ que l'on décrit, fonctionne par modes indexés par $i \in I$. Il cherche tour à tour à visiter B_1, puis B_2, etc... Puisque $Y \subseteq \widehat{Pre}_Y(B_i)$, il ne risque jamais de sortir de Y. De plus $Y \subseteq \bigwedge_{i \notin I} \overline{I}$ donc $\mathcal{U}$ satisfait $\Box Y \wedge \bigwedge_{i \notin I} \Box \overline{I}$ (la modalité $\Box$ et la conjonction commutent). Grâce à la propriété de l'attracteur fini, $\mathcal{U}$ parvient, avec probabilité 1 à visiter chaque B_i infiniment souvent, et ce presque sûrement.

Réciproquement, soit $\tau \in C'_I$, et $\mathcal{V}$ l'adversaire qui lui est associé. On définit $T \subseteq \mathsf{Conf}$ comme l'ensemble des configurations visitées avec probabilité positive dans le comportement de $\mathcal{V}$ à partir de τ :

$$T \stackrel{\text{def}}{=} \{\sigma | \Pr_{\mathcal{V}}(\tau \models \Diamond\sigma) > 0\}.$$

Clairement $\tau \in T$. De plus, $T \subseteq \bigwedge\limits_{i \notin I} \overline{A_I}$. Enfin, pour toutes configurations $\sigma \in T$ et pour tout indice $i \in I$, il existe un chemin $\sigma \xrightarrow{*} B_i$ qui ne peut à aucun moment sortir de T. Ceci tient au fait que $\mathcal{V}$ est à mémoire finie (cf. Proposition 4.12, page 79). Finalement, T est un post-point fixe :

$$T \subseteq \bigwedge_{i \notin I} \overline{A_i} \wedge \bigwedge_{i \in I} \widehat{Pre}^+_T(B_i).$$

On en déduit que T est inclus dans le plus grand point fixe, qui est l'union des post-points fixes. On conclut par $\tau \in T \subseteq \nu X. \bigwedge\limits_{i \notin I} \overline{A_i} \wedge \bigwedge\limits_{i \in I} \widehat{Pre}^+_X(B_i)$, et donc $C'_I \subseteq \nu X. \bigwedge\limits_{i \notin I} \overline{A_i} \wedge \bigwedge\limits_{i \in I} \widehat{Pre}^+_X(B_i)$.

Finalement $C'_I = \nu X. \bigwedge\limits_{i \notin I} \overline{A_i} \wedge \bigwedge\limits_{i \in I} \widehat{Pre}^+_X(B_i)$.

Dans un deuxième temps, on montre que C_I s'exprime à l'aide de C'_I. Plus précisément, on montre que les assertion suivantes sont équivalentes.

1. $\exists \mathcal{U}$ à mémoire finie, $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i \notin I} \Diamond\Box\overline{A_i} \wedge \bigwedge_{i \in I} \Box\Diamond B_i) = 1$,
2. $\exists \mathcal{V}$ à mémoire finie, $\Pr_{\mathcal{V}}(\sigma \models \Diamond C'_I) = 1$.

2. $\Longrightarrow$ 1. À partir de $\mathcal{V}$, on construit un adversaire $\mathcal{U}$ à mémoire finie qui satisfait la bonne propriété. Dans un premier temps, $\mathcal{U}$ imite $\mathcal{V}$ pour atteindre C'_I presque sûrement. Une fois dans C'_I (si c'est le cas), $\mathcal{U}$ se comporte comme l'adversaire décrit plus haut qui, à partir de toute configuration de C'_I permet de vérifier presque sûrement $\bigwedge_{i \in I} \Box\Diamond B_i \wedge \bigwedge_{i \notin I} \Box\overline{A_i}$. L'adversaire $\mathcal{U}$ est construit comme concaténation de deux adversaires à mémoire finie, il est donc lui même à mémoire finie. De plus il satisfait $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i \notin I} \Diamond\Box\overline{A_i} \wedge \bigwedge_{i \in I} \Box\Diamond B_i) = 1$.

1. $\Longrightarrow$ 2. On considère l'ensemble T des configurations visitées infiniment souvent presque sûrement : $T \stackrel{\text{def}}{=} \{\tau | \Pr_U(\sigma \models \Box\Diamond\tau) = 1\}$. Nécessairement, $T \subseteq \bigcap_{i \notin I} \overline{A_i}$ et pour tout indice $i \in I$, $T \cap B_i \neq \emptyset$. De plus, $\Pr_{\mathcal{U}}(\sigma \models \Diamond T) = 1$. Il reste à prouver que, pour toute configuration $\tau \in T$, il existe un adversaire $\mathcal{V}$ tel que $\Pr_{\mathcal{V}}(\tau \models \bigwedge_{i \notin I} \Box\overline{A_i} \wedge \bigwedge_{i \in I} \Box\Diamond B_i)$. En prenant l'adversaire suffixe de $\mathcal{U}$ (lorsque on ne quitte plus T), on obtient $\tau \models \bigwedge_{i \notin I} \Box\overline{A_i}$, puisque $T \subseteq \bigcap_{i \notin I} \overline{A_i}$. L'ensemble T est constitué de τ et de tous ses successeurs possibles, en une ou plusieurs étapes. Ceci est dû au fait que $\mathcal{U}$ est à mémoire finie. Comme cet ensemble intersecte chacun des B_i on obtient finalement :

$$\Pr_{\mathcal{V}}(\tau \models \bigwedge_{i \notin I} \overline{A_i} \wedge \bigwedge_{i \in I} \Box\Diamond B_i) = 1.$$

c'est-à-dire $\tau \in C'_I$. Clairement, l'ensemble des configurations visitées avec probabilité 1 sous $\mathcal{U}$ est accessible presque sûrement en utilisant $\mathcal{U}$, ce qui permet de conclure.

L'expression de C_I en fonction de C'_I repose enfin sur le Lemme 5.29 (cf. page 95), qui donne un terme de L_μ pour caractériser les configurations à partir desquelles il existe un adversaire (à mémoire finie) résolvant l'accessibilité presque sûrement. $\square$

Les ensembles C_I et C'_I peuvent être écrits sous la forme de termes gardés en utilisant la Proposition 5.22 (cf. page 93) :

$$\begin{aligned} C_I &= \nu Y.\widehat{Pre}^{*}_{K_\downarrow Y}(C'_I) \\ C'_I &= \nu X. \bigwedge_{i \notin I} \overline{A_i} \wedge \bigwedge_{i \in I} \widehat{Pre}^{+}_{K_\downarrow X}(B_i) \end{aligned}$$

Ainsi, si A_i et B_i sont des régions, les ensembles C_I et C'_I sont des régions effectivement calculables. On en déduit la décidabilité de (b).

Preuve de (c) Pour montrer la décidabilité de (c), on se ramène au cas précédent.

Lemme 5.54. *Soient $A_1, B_1, \cdots, A_n, B_n \subseteq \mathsf{Conf}$ des ensembles de configurations quelconques. Les assertions suivantes sont équivalentes :*

1. *$\exists \mathcal{U}$ à mémoire finie tel que $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{1 \le i \le n} \Box\Diamond A_i \Rightarrow \Box\Diamond B_i) > 0$,*
2. *$\sigma \xrightarrow{*} C = \bigcup_{I \subseteq \{1,\cdots,n\}} C_I$.*

Preuve : (2) $\Longrightarrow$ (1) est clair.

(1) $\Longrightarrow$ (2) On commence par remarquer que $\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{1 \le i \le n} \Box\Diamond A_i \Rightarrow \Box\Diamond B_i) > 0$ implique l'existence d'un sous-ensemble $I \subseteq \{1, \cdots, n\}$ tel que

$$\Pr_{\mathcal{U}}(\sigma \models \bigwedge_{i \notin I} \Diamond\Box\overline{A_i} \wedge \bigwedge_{i \in I} \Box\Diamond B_i) > 0$$

puisque le nombre de sous-ensembles de $\{1, \cdots, n\}$ est fini. On note $\phi_I \stackrel{\text{def}}{=} \bigwedge_{1 \le i \le n} \Box\Diamond A_i \Rightarrow \Box\Diamond B_i$. On utilise alors la propriété de l'attracteur fini pour exhiber une configuration τ et un mode u de $\mathcal{U}$ tels que

$$\Pr_{\mathcal{U}}(\sigma \models \phi_I \wedge \Box\Diamond \tau_u) > 0.$$

Visiter τ_u infiniment souvent assure de visiter tous les successeurs de τ_u (c'est-à-dire $Post^*(\tau_u)$) infiniment souvent aussi presque sûrement. En conséquence, l'ensemble des successeurs de τ_u est inclus dans $\bigcap_{i \notin I} \overline{A_i}$, et intersecte chacun des B_i. Ceci mis avec $\Pr_{\mathcal{U}}(\tau_u \models \bigwedge_{\tau' \in S(\tau_u)} \Box\Diamond\tau' \wedge \Box \bigvee_{\tau' \in S(\tau_u)} \tau') = 1$ entraîne

$$\Pr_{\mathcal{U}}(\tau_u \models \bigwedge_{i \notin I} \Box\overline{A_i} \wedge \bigwedge_{i \in I} \Box\Diamond B_i) = 1.$$

Alors $\tau \in C_I \subseteq C$, et C est accessible depuis σ. $\square$

En notant $\phi \stackrel{\text{def}}{=} \bigwedge_{1 \le i \le n} \Box\Diamond A_i \Rightarrow \Box\Diamond B_i$, le lemme précédent permet de caractériser les configurations à partir desquelles il existe un adversaire à mémoire finie vérifiant ϕ avec probabilité positive par le terme de L_μ :

$$\{\sigma \mid \exists \mathcal{U},\ \Pr_{\mathcal{U}}(\sigma \models \phi) > 0\} = \mu X.C \vee Pre(X) = \mu X.C \vee Pre(C_\uparrow X)$$

dans lequel X est gardé et C est calculable. Alors, en utilisant le Théorème 1.26 (cf. page 32) lorsque les $A_1, B_1, \cdots, A_n, B_n$ sont des régions, on obtient la décidabilité de (c).

Preuve de (d) Pour prouver la décidabilité de (d), on utilise la négation de la formule. Soit $\mathcal{U}$ un adversaire : alors $\Pr_{\mathcal{U}}(\sigma \models \phi) = 0$ si et seulement si $\Pr_{\mathcal{U}}(\sigma \models \neg\phi) = 1$. Dans notre cas, il suffit donc de considérer le problème de l'existence d'un adversaire à mémoire finie $\mathcal{U}$ tel que :

$$\Pr_{\mathcal{U}}\big(\sigma \models \bigvee_{1\leq i\leq n} (\Box\Diamond R_i \wedge \Diamond\Box\overline{R'_i})\big) = 1.$$

Définissons, pour $A_1, B_1, \cdots, A_n, B_n \subseteq \mathsf{Conf}$ des ensembles de configurations quelconques, et pour chaque $i \in \{1, \cdots, n\}$, l'ensemble de configurations

$$C_i \stackrel{\text{def}}{=} \{\tau \in \mathsf{Conf} | \exists \mathcal{V} \text{ à mémoire finie tel que } \Pr_{\mathcal{V}}(\tau \models \Box\Diamond A_i \wedge \Box\overline{B_i}) = 1\}.$$

On note $C \stackrel{\text{def}}{=} \bigcup_{1\leq i\leq n} C_i$. Remarquons que C_i est définissable par un terme gardé de L_μ

$$C_i = \nu X.\overline{B_i} \wedge \widehat{Pre}^{+}_{K_\downarrow X}(A_i).$$

On peut donc calculer effectivement les régions $(C_i)_{1\leq i\leq n}$ et la région C sous réserve que les $A_1, B_1, \cdots, A_n, B_n$ soient eux-mêmes des régions.

Le lemme qui suit réduit notre problème, à la question de l'accessibilité de C.

Lemme 5.55. *Les deux assertions suivantes sont équivalentes :*

1. *$\exists \mathcal{U}$ à mémoire finie tel que* $\Pr_{\mathcal{U}}\big(\sigma \models \bigvee_{1\leq i\leq n}(\Box\Diamond A_i \wedge \Diamond\Box\overline{B_i})\big) = 1$,
2. *$\exists \mathcal{V}$ à mémoire finie tel que* $\Pr_{\mathcal{V}}(\sigma \models \Diamond C) = 1$.

Preuve : L'implication $2 \Longrightarrow 1$ est simple, et basée sur la construction de $\mathcal{U}$ par combinaison de $\mathcal{V}$, pour atteindre presque sûrement C, avec un adversaire qui, une fois C atteint, permet de satisfaire la propriété $\bigvee_{1\leq i\leq n} \Box\Diamond A_i \wedge \Box\overline{B_i}$ presque sûrement.

$1 \Longrightarrow 2$: On montre plus précisément que cette implication est vraie en prenant $\mathcal{V} = \mathcal{U}$. Supposons par contradiction que $\Pr_{\mathcal{U}}(\sigma \models \Diamond C) < 1$. Alors, la probabilité de rester en dehors de C est positive : $\Pr_{\mathcal{U}}(\sigma \models \Box\overline{C}) > 0$. La propriété de l'attracteur fini et le fait que $\mathcal{U}$ soit à mémoire fini permettent d'exhiber une configuration τ et un mode u de $\mathcal{U}$ tels que

$$\Pr_{\mathcal{U}}(\sigma \models \Box\overline{C} \wedge \Box\Diamond\tau_u) > 0.$$

En notant $S(\tau_u)$ les configurations accessibles à partir de τ_u dans la chaîne de Markov engendrée par $\mathcal{N}$ et $\mathcal{U}$, on obtient

$$\Pr_{\mathcal{U}}(\tau_u \models \Box S(\tau_u) \wedge \bigwedge_{\tau' \in S(\tau_u)} \Box\Diamond\tau') = 1.$$

Or $S(\tau_u)$ est nécessairement d'intersection non vide avec chacun des B_i et d'intersection vide avec chacun des A_i. Alors

$$\Pr_{\mathcal{U}}(\tau_u \models \bigvee_{1\leq i\leq n} \Box\Diamond A_i \wedge \Diamond\Box B_i) = 0.$$

Et, comme $\Pr_{\mathcal{U}}(\sigma \models \Diamond\tau_u) > 0$, cela entraîne

$$\Pr_{\mathcal{U}}(\sigma \models \bigvee_{1\leq i\leq n} \Box\Diamond A_i \wedge \Diamond\Box B_i) < 1,$$

ce qui contredit l'hypothèse de départ (assertion 1.). On conclut donc que $\Pr(\sigma \models \Diamond C) = 1$. $\Box$

Si les $A_1, B_1, \cdots, A_n, B_n$ sont des régions, C est une région calculable. D'autre part, on peut décider de l'existence d'un adversaire (à mémoire finie) assurant $\Diamond R$ presque sûrement pour une région R donnée. Ces deux éléments prouvent la décidabilité de (d).

Ceci termine la preuve du Théorème 5.50. $\square$

5.3.2 Propriétés ω-régulières

Plus expressives que les propriétés de Streett, on s'intéresse à présent aux propriétés ω-régulières, c'est-à-dire exprimables par un automate fini de mots infinis (automate de Büchi, automate de Müller déterministe, etc...). Idéalement, et pour continuer sur la même idée que jusqu'ici, ce sont les régions régulières qui serviraient de propositions atomiques pour les formules du temps linéaire. Par souci de simplification, nous considérons ici[2] que les propositions atomiques sont en fait des régions de contrôle, c'est-à-dire des ensembles d'états de contrôle.

Les résultats que nous allons énoncer dans la suite, à savoir la décidabilité des quatre variantes de vérification qualitative de formules du temps linéaire pour les NPLCS, valent également dans le cas plus général où les propositions atomiques sont des régions générales. Cependant, la démonstration serait plus fastidieuse, et nécessiterait l'utilisation de LCS étendus, plutôt que de LCS habituels. C'est pourquoi nous donnons ici la preuve dans un cadre restreint.

Dans le reste de cette sous-section, fixons $\mathcal{N} = \langle S, \mathsf{C}, \mathsf{M}, \Delta, \lambda \rangle$ un NPLCS.

On suppose que les propriétés ω-régulières sont données sous la forme d'un automate déterministe avec condition d'acceptation de Streett sur l'alphabet S (l'ensemble des états de contrôle du NPLCS $\mathcal{N}$). D'autres formalismes pour exprimer des propriétés ω-régulières : automate de Büchi non déterministes, termes du μ-calcul, etc... sont possibles. Les traductions entre ces différents modèles sont bien connues maintenant. Pour de plus amples détails, nous renvoyons le lecteur à [GTW02].

Définition 5.56 (Automate de Streett). *Un* automate de Streett *est un n-uplet $\mathcal{A} = \langle Q, \Sigma, \rightarrow, q_0, Acc \rangle$ où*
- *Q est un ensemble fini d'états,*
- *Σ est un alphabet fini,*
- *$\rightarrow \subseteq Q \times \Sigma \times Q$ est la relation de transition,*
- *$q_0 \in Q$ est l'état initial, et*
- *$Acc = \{(Q_1, Q'_1), \cdots, (Q_n, Q'_n)\}$ est la condition d'acceptation (les Q_i, Q'_i sont des sous-ensembles de Q).*

Classiquement, on écrit $q \xrightarrow{a} q'$ si $(q, a, q') \in \rightarrow$. $\mathcal{A}$ est *déterministe* si pour tout état $q \in Q$ et toute lettre $a \in \Sigma$, il y a exactement un état $q' \in Q$ tel que $q \xrightarrow{a} q'$.

Intuitivement, la condition d'acceptation Acc est une condition d'équité forte : $\psi_{\mathcal{A}} \stackrel{\text{def}}{=} \bigwedge_{i=1}^{n} (\square\Diamond Q_i \Rightarrow \square\Diamond Q'_i)$.

Le *langage accepté* par $\mathcal{A}$, noté $L(\mathcal{A})$, est constitué de tous les mots infinis $a_0 a_1 a_2 \cdots \in \Sigma^\omega$ tels que l'exécution induite dans $\mathcal{A}$ (elle existe et est unique car $\mathcal{A}$ est déterministe) $q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \cdots$ est acceptée par $\mathcal{A}$, c'est-à-dire que pour tout indice $i \in \{1, \cdots, n\}$ soit il y a un nombre fini d'indices j tels que $q_j \in Q_i$, soit pour un nombre infini d'indices j, $q_j \in Q'_i$.

Considérons à présent un automate de Streett $\mathcal{A}$ sur l'alphabet S des états de contrôle de $\mathcal{N}$. Pour une exécution π de $\mathcal{N}$, on écrit $\pi \models \mathcal{A}$ si la projection de π sur S^ω est un mot de $L(\mathcal{A})$.

[2]comme nous l'avons fait aussi pour les PLCS

Les résultats positifs sur la vérification de propriétés de Streett pour les NPLCS entraînent alors :

Théorème 5.57 (Vérification de propriétés ω-régulières)**.** *Le problème de savoir, étant donné un NPLCS $\mathcal{N}$, une configuration σ et un automate de Streett $\mathcal{A}$ sur S, s'il existe un adversaire $\mathcal{U}$* à mémoire finie *tel que* $\Pr_{\mathcal{U}}(\sigma \models \mathcal{A}) = 1$ *(ou < 1 ou > 0 ou $= 0$), est décidable.*

Le passage de propriétés de Streett à des formules du temps linéaire suit la démarche classique, que nous avons déjà utilisée dans le cas des PLCS. On réduit la question de savoir si $\mathcal{N}$ est accepté par $\mathcal{A}$ à une question de vérification de propriété de Streett dans le produit $\mathcal{N} \times \mathcal{A}$ (voir [Var85] pour une présentation de cette méthode). Nous expliquons maintenant les étapes de cette réduction qui donnera la preuve du Théorème 5.57.

À partir de $\mathcal{N}$ et de $\mathcal{A}$, on construit leur produit : $\mathcal{N}' = \mathcal{N} \times \mathcal{A}$ qui est un NPLCS avec :
- les états de contrôle qui forment S' sont des paires (s, q) où $s \in S$ et $q \in Q$;
- l'ensemble des canaux C' est identique à C ;
- l'ensemble des messages M' est identique à M ;
- il y a une règle de transition $(s, q) \xrightarrow{\delta} (s', q')$ dans $\mathcal{N}'$ si et seulement si $s \xrightarrow{\delta} s'$ est une règle transition et $\mathcal{N}$ et $q \xrightarrow{s} q'$ dans $\mathcal{A}$.

Avec cette définition, chaque exécution π dans $\mathcal{N}$, de la forme

$$(s_0, \mathbf{w}_0) \to (s_1, \mathbf{w}_1) \to (s_2, \mathbf{w}_2) \to (s_2, \mathbf{w}_3) \cdots$$

donne lieu à une exécution π' dans $\mathcal{N}' = \mathcal{N} \times \mathcal{A}$

$$(s_0, q_0, \mathbf{w}_0) \to (s_1, q_1, \mathbf{w}_1) \to (s_2, q_2, \mathbf{w}_2) \to (s_3, q_3, \mathbf{w}_3) \cdots$$

où pour tout $j \in \mathbb{N}$, q_{j+1} et l'unique état de $\mathcal{A}$ vérifiant $q_j \xrightarrow{s_j} q_{j+1}$. Alors $q_0 \xrightarrow{s_0} q_1 \xrightarrow{s_1} q_2 \cdots$ est la seule exécution dans $\mathcal{A}$ qui correspond à π. Réciproquement, toute exécution dans $\mathcal{N} \times \mathcal{A}$ provient d'une exécution dans $\mathcal{N}$ et d'une exécution dans $\mathcal{A}$.

Supposons que la condition d'acceptation de $\mathcal{A}$ est donnée par $\psi_{\mathcal{A}} \stackrel{\text{def}}{=} \bigwedge_{i=1}^{n} (\Box\Diamond Q_i \Rightarrow \Box\Diamond Q'_i)$, c'est-à-dire que $Acc = \{(Q_1, Q'_1), \cdots, (Q_n, Q'_n)\}$. Alors on définit $R_i = S \times Q_i$ et $R'_i = S \times Q'_i$, et on considère dans $\mathcal{N} \times \mathcal{A}$ la condition de Streett suivante :

$$\psi_{\mathcal{N}\times\mathcal{A}} \stackrel{\text{def}}{=} \bigwedge_{i=1}^{n} (\Box\Diamond R_i \Rightarrow \Box\Diamond R'_i).$$

Lemme 5.58. *Soit π une exécution de $\mathcal{N}$ et π' l'exécution correspondante dans $\mathcal{N}'$. Alors*

$$\pi \models \mathcal{A} \quad ssi \quad \pi' \models \psi_{\mathcal{N}\times\mathcal{A}}.$$

Mieux, la bijection entre les exécutions de $\mathcal{N}$ et celles de $\mathcal{N} \times \mathcal{A}$ transforme un adversaire $\mathcal{U}$ pour $\mathcal{N}$ en un adversaire $\mathcal{V}$ pour $\mathcal{N} \times \mathcal{A}$ avec les mêmes probabilités, et vice et versa. Précisément :

Lemme 5.59. *Soit $p \in [0, 1]$. Alors, les assertions suivantes sont équivalentes :*
1. *Il existe un adversaire à mémoire finie $\mathcal{U}$ pour $\mathcal{N}$ tel que* $\Pr_{\mathcal{U}}((s, \varepsilon) \models \mathcal{A}) = p$.
2. *Il existe un adversaire à mémoire finie $\mathcal{V}$ pour $\mathcal{N}'$ tel que* $\Pr_{\mathcal{V}}((s, q_0, \varepsilon) \models \psi_{\mathcal{N}\times\mathcal{A}}) = p$.

Le Lemme 5.58 permet donc de réduire la vérification qualitative de propriétés ω-régulières sur $\mathcal{N}$ à la vérification qualitative de formules de Streett sur $\mathcal{N}'$. Grâce à ce lemme, le Théorème 5.57 devient une conséquence du Théorème 5.50 (cf. page 108).

5.4 Adversaires équitables

On s'intéresse maintenant au problème de la vérification qualitative de propriétés sous des hypothèses d'équité. Dans ce but, on considère une notion d'adversaire équitable qui permet de ne considérer que les adversaires satisfaisant presque sûrement une hypothèse d'équité donnée. Cela permet d'éliminer les adversaires qui engendrent un ensemble d'exécutions non équitables de mesure positive. Nous avons introduit la notion d'adversaire équitable et énoncé les résultats les concernant dans [BBS06c].

Remarque 5.60. *Cette notion d'équité pour les adversaires n'est pas liée aux notions d'*extreme-fairness *ou d'*alpha-fairness *étudiées dans [PZ86]. Ces dernières imposent que les choix probabilistes soient résolus de façon « équitable » et constituent des outils pour certaines méthodes de vérification. Au contraire, nous introduisons ici une notion d'équité sur les choix non déterministes que les adversaires sont amenés à faire.*

Notre notion d'équité impose une équité forte pour les adversaires, c'est-à-dire un traitement équitable des choix non déterministes entre les actions. Cette notion est introduite par Hart, Sharir et Pnueli [HSP83] et Vardi [Var85] puis reprise dans le cadre plus général des systèmes probabilistes concurrents par Baier et Kwiatkowska [BK98]. Néanmoins, dans [HSP83], seule l'équité entre les processus est considérée, et pas entre toutes les actions possibles : une exécution π est dite équitable si tous les processus (dont la composition forme le système considéré) agissent infiniment souvent le long de π. Dans notre définition, des propriétés d'équité plus générales que celle entre les processus peuvent être exprimées. De plus, à la différence de la définition dans [BK98], nous pouvons restreindre les contraintes d'équité à des sous-ensembles de l'ensemble des actions Δ.

Un adversaire est dit *équitable* si les exécutions qu'il engendre sont presque sûrement équitables, conformément à une contrainte d'équité donnée. On considère dans notre étude des conditions d'*équité forte* sur les règles de transitions choisies par l'adversaire. On suppose donc que la contrainte d'équité est donnée par un ensemble $\mathcal{F} = \{f_1, \cdots, f_k\}$, où chaque $f_i \subseteq \Delta$ est un sous-ensemble de règles de transitions, et on impose une équité forte pour chacun des f_i.

Exemple 5.61. *L'équité entre k processus $P_1, \cdots, P_k$ peut s'exprimer par le biais d'hypothèse d'équité sur les adversaires. Il suffit en effet de considérer $\mathcal{F} = \{f_1, \cdots, f_k\}$ où f_i est constitué des règles de transitions concernant le processus P_i.*

Un ensemble de règles $f \subseteq \Delta$ est dit *tirable* dans la configuration σ s'il existe une transition $\delta \in f$ qui est elle-même tirable dans σ. Si $F \subseteq \mathcal{F}$ est un sous-ensemble de l'ensemble des hypothèses d'équité, F est tirable dans σ s'il existe $f \in F$ tirable dans σ.

Définition 5.62 (Exécutions équitables, Adversaire équitables)**.** *Soit $\mathcal{F} \in 2^{2^\Delta}$ un ensemble de sous-ensembles de Δ.*

- *Une exécution infinie $\sigma_0 \xrightarrow{\delta_1} \sigma_1 \xrightarrow{\delta_2} \cdots$ est dite* équitable *pour $\mathcal{F}$ si et seulement si, pour tout $f \in \mathcal{F}$, soit $\delta_j \in f$ pour une infinité d'indices j, soit il existe un indice $i \geq 0$ pour lequel f n'est plus tirable dans les configurations σ_j avec $j \geq i$.*
- *L'adversaire $\mathcal{U}$ est* équitable *pour $\mathcal{F}$ (ou simplement* équitable *si $\mathcal{F}$ est clair dans le contexte) si quelle que soit la configuration initiale, presque toutes les exécutions sont équitables pour $\mathcal{F}$.*

Une question naturelle est la réalisabilité de tels adversaires : à mémoire finie et équitables. La proposition suivante répond positivement à cette question, quelque soit la contrainte d'équité.

Proposition 5.63. *Pour tout $\mathcal{F} \subseteq 2^{\Delta}$, il existe un adversaire à mémoire finie qui est équitable pour $\mathcal{F}$.*

Preuve : Décrivons un adversaire à mémoire finie, équitable pour $\mathcal{F}$ et ne vérifiant pas de propriété particulière supplémentaire. On construit $\mathcal{U}$ dont les modes sont des permutations $(f_1, \cdots, f_k)$ des éléments de $\mathcal{F}$. Le mode initial est arbitraire. Lorsque le mode est $(f_1, \cdots, f_k)$ et dans la configuration σ, le choix de $\mathcal{U}$ est le suivant. Soit $F = \{f \in \mathcal{F} | f \cap \Delta(\sigma) \neq \emptyset\}$ l'ensemble des hypothèses d'équité qui sont tirables dans σ. Si F est vide, $\mathcal{U}$ choisit de façon arbitraire une règle de transition tirable dans σ, et conserve le mode $(f_1, \cdots, f_k)$. Si $F \neq \emptyset$ et que i est l'indice minimal j tel que $f_j \in F$, $\mathcal{U}$ choisit une transition de $f_i \cap \Delta(\sigma)$ et passe en mode $(f_1, \cdots, f_{i-1}, f_{i+1}, \cdots, f_k, f_i)$. Par cette construction, $\mathcal{U}$ est un adversaire à mémoire finie pour lequel toutes (et donc presque toutes) les exécutions sont équitables pour $\mathcal{F}$. A fortiori, $\mathcal{U}$ est équitable pour $\mathcal{F}$. □

Pour certaines propriétés, les hypothèses d'équité ne sont pas pertinentes. C'est par exemple le cas des propriétés d'accessibilité $\Diamond A$ avec probabilité positive ou presque sûrement. En effet, si l'ensemble A est atteint, l'adversaire peut ensuite adopter un comportement équitable, en imitant l'adversaire décrit dans la preuve de la Proposition 5.63. Ceci est dû au fait que les propriétés $\Diamond A$ avec probabilité positive ou probabilité 1 ne concernent pas le comportement à long terme de l'adversaire.

Lemme 5.64. *Soit $\mathcal{F}$ une hypothèse d'équité. Alors,*

$$\exists \mathcal{U} \text{ équitable pour } \mathcal{F} \text{ tel que } \Pr_{\mathcal{U}}(\sigma \models \Diamond A) > 0 \quad \text{ssi} \quad \exists \mathcal{U} \text{ tel que } \Pr_{\mathcal{U}}(\sigma \models \Diamond A) > 0,$$
$$\exists \mathcal{U} \text{ équitable pour } \mathcal{F} \text{ tel que } \Pr_{\mathcal{U}}(\sigma \models \Diamond A) = 1 \quad \text{ssi} \quad \exists \mathcal{U} \text{ tel que } \Pr_{\mathcal{U}}(\sigma \models \Diamond A) = 1$$

Preuve : Dans les deux cas, l'implication de gauche à droite est triviale. Montrons donc l'autre implication.

Supposons qu'il existe un adversaire $\mathcal{U}$ vérifiant $\Pr_{\mathcal{U}}(\sigma \models \Diamond A) > 0$. On choisit un chemin simple qui suit le comportement de $\mathcal{U}$ et mène de σ à A : $\pi : \sigma \xrightarrow{*} \tau \in A$. Soit alors $\mathcal{V}$ l'adversaire qui imite $\mathcal{U}$ pour tout préfixe strict de π, et qui, une fois τ atteint (ce qui arrive avec probabilité positive), se comporte de façon équitable pour $\mathcal{F}$ (voir Proposition 5.63). L'adversaire $\mathcal{V}$ ainsi construit est équitable pour $\mathcal{F}$ et satisfait $\Pr_{\mathcal{V}}(\sigma \models \Diamond A) > 0$. De plus, si $\mathcal{U}$ est à mémoire finie, c'est encore le cas pour $\mathcal{V}$.

Supposons à présent que $\mathcal{U}$ est un adversaire satisfaisant $\Pr_{\mathcal{U}}(\sigma \models \Diamond A) = 1$. On définit $\mathcal{V}$ comme étant un adversaire qui, pour tous les chemins finis qui ne visitent pas A, fait les mêmes choix que $\mathcal{U}$, et une fois A atteint (ce qui arrive presque sûrement) se comporte équitablement pour $\mathcal{F}$, là encore grâce à la Proposition 5.63. Cet adversaire est équitable pour $\mathcal{F}$ et vérifie $\Pr_{\mathcal{V}}(\sigma \models \Diamond A) = 1$. Ici encore, si $\mathcal{U}$ est à mémoire finie, $\mathcal{V}$ l'est également. □

On en déduit la décidabilité du problème, étant donné une configuration σ, une *région* R, et une hypothèse d'équité $\mathcal{F}$, de savoir s'il existe un adversaire équitable pour $\mathcal{F}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Diamond R) > 0$ (resp. $= 1$).

Théorème 5.65. *On peut décider, étant donnés un NPLCS $\mathcal{N}$, une configuration σ, une* région $R \subseteq \mathsf{Conf}$ *et une hypothèse d'équité $\mathcal{F} \subseteq 2^{\Delta}$, s'il existe un adversaire $\mathcal{U}$ équitable pour $\mathcal{F}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Diamond R) > 0$ (resp. $= 1$).*

Preuve : Le Théorème 5.65 est une conséquence directe de la Proposition 5.63, et du Théorème 5.31 (cf. page 96). □

Contrairement aux questions d'accessibilité, les propriétés d'invariant $\Box A$ avec probabilité positive ou $= 1$ ne peuvent pas se réduire aux même problèmes pour des adversaires non équitables. Il faut pour montrer la décidabilité de telles questions, généraliser les équations de points fixes données précédemment pour prendre en compte les hypothèses d'équité. On définit :

$$I_\emptyset(\mathcal{F}, A) \stackrel{\text{def}}{=} \nu X.\big(A \setminus \mathit{Pre}[\mathcal{F}](\mathsf{Conf})\big) \wedge \widehat{\mathit{Pre}}_X(\mathsf{Conf})$$
$$I_F(\mathcal{F}, A) \stackrel{\text{def}}{=} \nu X.\big(A \setminus \mathit{Pre}[\mathcal{F} \setminus F](\mathsf{Conf})\big) \wedge \bigwedge_{f \in F} \widehat{\mathit{Pre}}^*_X\big(\widehat{\mathit{Pre}}_X[f](\mathsf{Conf})\big)$$
$$I(\mathcal{F}, A) \stackrel{\text{def}}{=} \bigvee_{F \subset \mathcal{F}} I_F(\mathcal{F}, A)$$

Remarque 5.66. *Cette série de définition est bien fondée : elle se base sur une induction sur* $|\mathcal{F}|$.

Chacun des $I_F(\mathcal{F}, A)$ (même lorsque $F = \emptyset$) peut être vu comme un terme gardé grâce à la Proposition 5.22 (cf. page 93) et $I(\mathcal{F}, A)$ vérifie la propriété :

Lemme 5.67. *Il existe un adversaire $\mathcal{U}$ sans mémoire et équitable pour $\mathcal{F}$ tel que pour toute configuration $\sigma \in I(\mathcal{F}, A)$,*

$$\Pr_{\mathcal{U}}(\sigma \models \Box A) = 1.$$

Preuve : On commence par montrer que pour tout sous-ensemble $F \subseteq \mathcal{F}$, il existe un adversaire à mémoire finie et équitable pour $\mathcal{F}$, noté $\mathcal{U}_F$ tel que pour toute configuration $\sigma \in I_F(\mathcal{F}, A)$, $\Pr_{\mathcal{U}_F}(\sigma \models \Box A) = 1$.

Si $F = \emptyset$, ceci est clair et on peut même choisir un adversaire sans mémoire $\mathcal{U}_\emptyset$ tel que $\Pr_{\mathcal{U}_\emptyset}(\sigma \models \Box A) = 1$.

Soit F un sous-ensemble non vide de $\mathcal{F}$: $F = \{f_1, \cdots, f_k\}$. Pour toute configuration $\sigma \in I_F(\mathcal{F}, A)$, et tout $f \in F$, on choisit un chemin fini de la forme

$$\pi_{\sigma,f} : \ \sigma \stackrel{\text{def}}{=} \sigma_0 \xrightarrow{\delta_1} \sigma_1 \cdots \xrightarrow{\delta_m} \sigma_m$$

avec $m \geq 0$, $\delta_m \in f$, et pour tout $1 \leq i \leq m$, $\mathit{Post}[\delta_i](\sigma_{i-1}) \subseteq I_F(\mathcal{F}, A)$, et pour tout $0 \leq i \leq m$, $\sigma_i \in A \setminus \mathit{Pre}[\mathcal{F}](\mathsf{Conf})$. On suppose également que δ_m est la première transition à appartenir à f. On fait également l'hypothèse que $\pi_{\sigma_i,f}$ coïncide avec le suffixe de $\pi_{\sigma,f}$ partant de σ_i.

Décrivons un adversaire à mémoire finie $\mathcal{U}_F$ dont les modes sont les hypothèses d'équité $f \in F$. Dans la configuration $\sigma \in I_F(\mathcal{F}, A)$, et en mode $f \in F$, $\mathcal{U}_F$ cherche à concrétiser le chemin $\pi_{\sigma,f}$. Il y parvient avec probabilité positive. Dans tous les cas, le chemin reste dans $I_F(\mathcal{F}, A)$ par hypothèse. Si une transition $\delta_m \in f$ est tirée, l'adversaire passe en mode $f_{(j+1) \bmod k}$ si $f = f_j$. La propriété de l'attracteur fini assure que presque sûrement une transition de f sera exécutée. Les exécutions engendrées par $\mathcal{U}_F$ sont presque sûrement des exécutions équitables pour $\mathcal{F}$, puisque, avec probabilité 1, pour tout $f \in F$ infiniment souvent une transition $\delta \in f$ est prise et les autres ensembles d'équité $g \in \mathcal{F} \setminus F$ ne sont pas tirables dans les configurations de $I_F(\mathcal{F}, A)$. Ainsi $\mathcal{U}_F$ est à mémoire finie et équitable pour $\mathcal{F}$; de plus, comme $I_F(\mathcal{F}, A) \subseteq A$, $\mathcal{U}_F$ satisfait $\Box A$. A fortiori $\Pr_{\mathcal{U}_F}(\sigma \models \Box A) = 1$, pour toute configuration $\sigma \in I(\mathcal{F}, A)$.

Il reste maintenant à composer les différents adversaires $\mathcal{U}_F$ pour $F \subseteq \mathcal{F}$. Soit $F_1, \cdots, F_m$ une énumération des sous-ensembles de $\mathcal{F}$, avec la contrainte que $F_i \subset F_j$ implique $i \leq j$. On a donc $F_0 = \emptyset$ et $F_m = \mathcal{F}$. On définit alors des ensembles de configurations T_i, pour $1 \leq i \leq m$:

$$
\begin{aligned}
&T_1 \stackrel{\text{def}}{=} I(\mathcal{F}, A), \\
&T_i \stackrel{\text{def}}{=} I_{F_i}(\mathcal{F}, A) \setminus (T_1 \cup \cdots \cup T_{i-1}) \qquad\qquad \text{si } i > 0.
\end{aligned}
$$

De cette façon, les $(T_i)_{1 \leq i \leq m}$ forment une partition de $I(\mathcal{F}, A)$. À chaque configuration $\sigma \in T_i$ et $f \in F_i$, on choisit $\pi_{\sigma,f}$ comme précédemment. Alors

$$
Post[\delta_i](\sigma_i) \in T_1 \cup \cdots \cup T_i \text{ pour tout } 1 \leq i \leq m
$$

$\mathcal{U}$ est défini comme un adversaire à mémoire finie, ayant des modes (i, f) avec $1 \leq i \leq m$ et $f \in F_i$. Dans la configuration initiale σ_0, $\mathcal{U}$ est en mode i tel que $\sigma_0 \in T_i$. Lorsqu'il est en mode (i, f) et dans la configuration σ, $\mathcal{U}$ se comporte comme $\mathcal{U}_{F_i}$ en mode f et en configuration σ, c'est-à-dire que $\mathcal{U}$ essaie d'engendrer le chemin $\pi_{\sigma,f}$. Tant que la configuration courante est dans T_i, $\mathcal{U}$ continue à simuler $\mathcal{U}_{F_i}$. Dès qu'une configuration $\sigma' \in T_j$ pour $j < i$ est atteinte, $\mathcal{U}$ change de mode pour (j, f') avec $f' \in F_j$ arbitraire, et imite alors le comportement de $\mathcal{U}_{F_j}$. L'adversaire $\mathcal{U}$ ainsi obtenu est à mémoire finie, équitable pour $\mathcal{F}$ et satisfait $\Pr_{\mathcal{U}}(\sigma \models \Box A) = 1$ pour tout $\sigma \in I(\mathcal{F}, A)$. $\square$

Avant de passer à la vérification d'invariants avec probabilité positive, nous introduisons les notions d'équité probabiliste et d'équité forte probabiliste.

Définition 5.68 (Équité (forte) probabiliste)**.** *Soit π une exécution. On note $\mathtt{Inf}(\pi)$ l'ensemble des configurations visitées infiniment souvent dans π. L'exécution π est dite :*
- équitable vis à vis des probabilités *abrégé en* prob-équitable *si $\mathtt{Inf}(\pi) \neq \emptyset$ et pour toute configuration $\sigma \in \mathtt{Inf}(\pi)$ et toute règle de transition $\delta \in \Delta(\sigma)$ tirée infiniment souvent dans π, toutes les configurations $\tau \in \delta(\sigma)$ apparaissent infiniment souvent dans π ;*
- fortement prob-équitable *si π est prob-équitable, et il existe un suffixe infini π' de π tel que tout sous-chemin fini de π' y apparaît infiniment souvent (et donc infiniment souvent dans π).*

Remarque 5.69. *Pour les adversaires quelconques, presque toutes les exécutions sont prob-équitables et pour les adversaire à mémoire finie, presque toutes les exécutions sont fortement prob-équitables.*

Lemme 5.70. *Si π est une exécution équitable pour $\mathcal{F}$ et fortement prob-équitable vérifiant $\pi \models \Box A$, alors $\mathtt{Inf}(\pi) \subseteq I(\mathcal{F}, A)$.*

Preuve : Puisque π est équitable pour $\mathcal{F}$, il existe un ensemble $F \subseteq \mathcal{F}$ tel que tout $f \in \mathcal{F} \setminus F$ est tirable seulement un nombre fini de fois, alors que pour tout $f \in F$, il existe une règle $\delta \in f$ tirée infiniment souvent le long de π. Montrons que $\mathtt{Inf}(\pi)$ satisfait les conditions de la caractérisation par point fixe de $I_F(\mathcal{F}, A)$. Ainsi, $\mathtt{Inf}(\pi)$ sera un post point fixe, et sera donc un sous-ensemble du plus grand point fixe, c'est-à-dire $I_F(\mathcal{F}, A)$.

Commençons par le cas particulier $F = \emptyset$. Rappelons que $I_\emptyset(\mathcal{F}, A) = \nu X.\big(A \setminus Pre[\mathcal{F}](\mathsf{Conf})\big) \wedge \widehat{Pre}_X(\mathsf{Conf})$. Par définition de F, et puisque $\pi \models \Box A$, on obtient $\mathtt{Inf}(\pi) \subseteq A \setminus Pre[\mathcal{F}](\mathsf{Conf})$. En effet $Pre[\mathcal{F}](\mathsf{Conf})$ représente l'ensemble des configurations où $\mathcal{F}$ est tirable. De plus, si $\tau \in \mathtt{Inf}(\pi)$ et $\delta \in \Delta(\tau)$ est une transition tirée infiniment souvent au cours de π, on doit avoir

$Post[\delta](\tau) \subseteq \texttt{Inf}(\pi)$ puisque π est prob-équitable. Ce qui montre bien que $\texttt{Inf}(\pi)$ est un post point fixe de $X \to \big(A \setminus Pre[\mathcal{F}](\mathsf{Conf})\big) \wedge \widehat{Pre}_X(\mathsf{Conf})$.

Supposons à présent que F est non vide. On a bien $\texttt{Inf}(\pi) \subseteq A \setminus Pre[\mathcal{F} \setminus F](\mathsf{Conf})$ par définition de F et puisque $\pi \models \Box A$. Soit $\tau \in \texttt{Inf}(\pi)$ et $f \in F$. Puisque π est fortement prob-équitable, il existe un sous-chemin de π :

$$\tau = \sigma_0 \xrightarrow{\delta_1} \sigma_1 \cdots \xrightarrow{\delta_m} \sigma_m \text{ avec } \delta_m \in f$$

qui apparaît infiniment souvent dans π. Puisque $\pi \models \Box A$ et π est prob-équitable, on obtient $Post[\delta_i](\sigma_{i-1}) \subseteq \texttt{Inf}(\pi)$ pour tout $1 \leq i \leq m$. Ainsi, $\texttt{Inf}(\pi)$ est un post point fixe de $X \to \big(A \setminus Pre[\mathcal{F} \setminus F](\mathsf{Conf})\big) \wedge \bigwedge_{f \in F} \widehat{Pre}^*_X\big(\widehat{Pre}_X[f](\mathsf{Conf})\big)$.

Puisque les post-points fixes sont des sous-ensembles des plus grands points fixes, dans les deux cas ($F = \emptyset$ et $F \neq \emptyset$) on a montré que $\texttt{Inf}(\pi) \subseteq I_F(\mathcal{F}, A) \subseteq I(\mathcal{F}, A)$. $\square$

Théorème 5.71. *Soit $\mathcal{N}$ un NPLCS, $A \subseteq \mathsf{Conf}$ un ensemble de configurations, et $\mathcal{F}$ une contrainte d'équité.*

L'ensemble des configurations pour lesquelles il existe un adversaire $\mathcal{U}$ à mémoire finie équitable pour $\mathcal{F}$ et tel que $\Pr_{\mathcal{U}}(\sigma \models \Box A) > 0$ est caractérisé par le terme de L_μ :

$$\mu Y.\ \ I(\mathcal{F}, A) \vee \big(A \wedge Pre(Y)\big).$$

Preuve : Soient σ une configuration et $\mathcal{V}$ un adversaire équitable pour $\mathcal{F}$ et à mémoire finie satisfaisant $\Pr_{\mathcal{V}}(\sigma \models \Box A) > 0$. On considère une exécution π conforme à $\mathcal{V}$, qui soit équitable pour $\mathcal{F}$, fortement prob-équitable et vérifiant $\pi \models \Box A$. Une telle exécution existe puisque presque toutes les exécutions conformes à $\mathcal{V}$ sont équitables pour $\mathcal{F}$ et fortement prob-équitables. D'après le Lemme 5.70, $\texttt{Inf}(\pi) \subseteq I(\mathcal{F}, A)$ et donc $\sigma \models \exists(A\ Until\ I(\mathcal{F}, A))$, c'est-à-dire, $\sigma \in \mu Y.\ \ I(\mathcal{F}, A) \vee \big(A \wedge Pre(Y)\big)$.

Réciproquement, supposons que $\sigma \models \exists(A\ Until\ I(\mathcal{F}, A))$. Alors il existe $F \subseteq \mathcal{F}$ tel que $\sigma \models \exists(A\ Until\ I_F(\mathcal{F}, A))$. Soit π un chemin témoin :

$$\pi :\ \sigma = \sigma_0 \xrightarrow{\delta_1} \sigma_1 \cdots \xrightarrow{\delta_m} \sigma_m \in I_F(\mathcal{F}, A)$$

avec $\sigma_i \in A$ pour $0 \leq i \leq m-1$. On construit un adversaire $\mathcal{V}$ qui cherche à réaliser ce chemin. S'il échoue, c'est-à-dire si les pertes ne sont pas celles espérées, $\mathcal{V}$ se comporte alors de façon arbitraire, mais équitable pour $\mathcal{F}$. Si $I_F(\mathcal{F}, A)$ et donc $I(\mathcal{F}, A)$ est atteint, $\mathcal{V}$ se comporte alors comme l'adversaire décrit dans la preuve du Lemme 5.67 (cf. page 118), c'est-à-dire qu'il vérifie $\Box A$ presque sûrement une fois $I(\mathcal{F}, A)$ atteint. L'adversaire $\mathcal{V}$ ainsi construit est à mémoire finie, équitable pour $\mathcal{F}$ et satisfait $\Pr \mathcal{V}(\sigma \models \Box A) > 0$. $\square$

Le Lemme 5.67 (cf. page 118) n'est pas exactement une caractérisation de l'ensemble des configurations à partir desquelles il existe un adversaire à mémoire finie et équitable pour $\mathcal{F}$ permettant d'avoir $\Box R$ presque sûrement. En effet, il ne donne qu'une implication. Dans le théorème qui suit, on donne une véritable caractérisation.

Théorème 5.72. *Soit $\mathcal{N}$ un NPLCS, $A \subseteq \mathsf{Conf}$ un ensemble de configurations, et $\mathcal{F}$ une contrainte d'équité.*

L'ensemble des configurations pour lesquelles il existe un adversaire $\mathcal{U}$ à mémoire finie équitable pour $\mathcal{F}$ et tel que $\Pr_{\mathcal{U}}(\sigma \models \Box A) = 1$ est caractérisé par le terme de L_μ :

$$\nu X.\ \ \widehat{Pre}^*_X(I(\mathcal{F}, A)) \wedge (A \vee I(\mathcal{F}, A)).$$

Preuve : On commence par observer l'équivalence suivante : étant donnés $A, B \subseteq \mathsf{Conf}$,

$$\sigma \in \nu X.\ \widehat{Pre}^*_X(B) \wedge (A \vee B) \text{ ssi } \exists \mathcal{U} \text{ sans mémoire } \Pr_{\mathcal{U}}(\sigma \models A \text{ } Until \text{ } B) = 1.$$

Soient $\sigma \in \mathsf{Conf}$ et $\mathcal{V}$ un adversaire à mémoire finie et équitable pour $\mathcal{F}$ tel que $\Pr_{\mathcal{V}}(\sigma \Box A) = 1$. On considère l'ensemble Π des exécutions conformes à $\mathcal{V}$, équitables pour $\mathcal{F}$, fortement prob-équitables et satisfaisant $\Box A$. Cet ensemble est non vide par hypothèses sur $\mathcal{V}$. Soit alors $T \stackrel{\text{def}}{=} \bigcup_{\pi \in \Pi} \mathtt{Inf}(\pi)$. Le Lemme 5.70 (cf. page 119) implique que $T \subseteq I(\mathcal{F}, A)$. De plus, puisque $\mathcal{V}$ est à mémoire finie, l'ensemble des exécutions équitables pour $\mathcal{F}$ et fortement prob-équitable est de mesure 1. On en conclut que $\Pr_{\mathcal{V}}(\sigma \models A \text{ } Until \text{ } I(\mathcal{F}, A)) = 1$. Et donc $\sigma \in \nu X.\ \widehat{Pre}^*_X(I(\mathcal{F}, A)) \wedge (A \vee I(\mathcal{F}, A))$.

Soit à présent $\sigma \in \nu X.\ \widehat{Pre}^*_X(I(\mathcal{F}, A)) \wedge (A \vee I(\mathcal{F}, A))$. On note $B \stackrel{\text{def}}{=} I(\mathcal{F}, A)$ pour faciliter la lecture. Alors il existe un adversaire à mémoire finie $\mathcal{U}$ tel que $\Pr_{\mathcal{U}}(\sigma \models A \text{ } Until \text{ } B) = 1$. On modifie alors $\mathcal{U}$ en un adversaire $\mathcal{V}$ qui soit équitable pour $\mathcal{F}$ et à mémoire finie tel que $\Pr_V(\sigma \models \Box A) = 1$. $\mathcal{V}$ imite $\mathcal{U}$ tant que B n'est pas encore atteint. Une fois que B est atteint, $\mathcal{V}$ se comporte comme un adversaire à mémoire finie, équitable pour $\mathcal{F}$ satisfaisant $\Box A$ presque sûrement (comme on l'a vu dans la preuve du Lemme 5.67, page 118). L'adversaire $\mathcal{V}$ ainsi construit est à mémoire finie, équitable pour $\mathcal{F}$ et remplit la condition : $\Pr_{\mathcal{V}}(\sigma \models \Box A) = 1$. $\square$

Corollaire 5.73. *Soit $\mathcal{N}$ un NPLCS, R une* région*, et $\mathcal{F}$ une hypothèse d'équité.*

On peut calculer l'ensemble des configurations σ telles qu'il existe un adversaire $\mathcal{U}$ à mémoire finie et équitable pour $\mathcal{F}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Box R) > 0$.

Les ensembles suivants sont calculables et sont des régions :
- $\{\sigma \mid \exists \mathcal{U}$ *à mémoire finie, et équitable pour* $\mathcal{F}$ *t.q.* $\Pr_{\mathcal{U}}(\sigma \models \Box R) > 0\}$
- $\{\sigma \mid \exists \mathcal{U}$ *à mémoire finie, et équitable pour* $\mathcal{F}$ *t.q.* $\Pr_{\mathcal{U}}(\sigma \models \Box R) = 1\}$

Preuve : Ce sont des conséquences des Théorèmes 5.71 et 5.72 et du Théorème 1.26 sur les termes gardés. $\square$

Dans la suite, on note $E^{\mathcal{F}}_{\Box A}$ l'ensemble des configurations σ telles qu'il existe un adversaire à mémoire finie et équitable pour $\mathcal{F}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Box A) = 1$.

On s'intéresse à des questions d'accessibilité répétée ($\Box\Diamond A$) et de façon duale, de persistance ($\Diamond\Box A$), sous des hypothèses d'équité.

Pour tout sous-ensemble $F \subseteq \mathcal{F}$, on définit :

$$E^F_{\Box\Diamond A} \stackrel{\text{def}}{=} \nu X. \Big(\mathsf{Conf} \setminus \big(Pre[\mathcal{F} \setminus F](\mathsf{Conf})\big)\Big) \wedge \widehat{Pre}^+_X(A) \wedge \bigwedge_{f \in F} \widehat{Pre}^*_X\big(\widehat{Pre}_X[f](\mathsf{Conf})\big).$$

Puis $E^{\mathcal{F}}_{\Box\Diamond A}$ comme l'union des $E^F_{\Box\Diamond A}$:

$$E^{\mathcal{F}}_{\Box\Diamond A} \stackrel{\text{def}}{=} \bigvee_{F \subseteq \mathcal{F}} E^F_{\Box\Diamond A}.$$

Théorème 5.74. *Soient $A \subseteq \mathsf{Conf}$ et $\sigma \in \mathsf{Conf}$.*

(a) *Il existe un adversaire $\mathcal{U}$ à mémoire finie, et équitable pour $\mathcal{F}$ tel que $\Pr_U(\sigma \models \Box\Diamond A) = 1$ si et seulement si $\sigma \in \nu X.\widehat{Pre}^*_X(E^{\mathcal{F}}_{\Box\Diamond A})$.*

(b) *Il existe un adversaire $\mathcal{U}$ à mémoire finie, et équitable pour $\mathcal{F}$ tel que $\Pr_U(\sigma \models \Box\Diamond A) > 0$ si et seulement si $\sigma \in Pre^*(E^{\mathcal{F}}_{\Box\Diamond A}))$.*

(c) *Il existe un adversaire $\mathcal{U}$ à mémoire finie, et équitable pour $\mathcal{F}$ tel que* $\Pr_U(\sigma \models \Diamond\Box A) = 1$ *si et seulement si* $\sigma \in \nu X.\widehat{Pre}^*_X(E^{\mathcal{F}}_{\Box A})$.

(d) *Il existe un adversaire $\mathcal{U}$ à mémoire finie, et équitable pour $\mathcal{F}$ tel que* $\Pr_U(\sigma \models \Diamond\Box A) > 0$ *si et seulement si* $\sigma \in Pre^*(E^{\mathcal{F}}_{\Box A})$.

Preuve : Les preuves des assertions (a) à (d) reposent sur les observations suivantes :

1. Il existe un adversaire $\mathcal{W}$ à mémoire finie équitable pour $\mathcal{F}$ tel que $\Pr_{\mathcal{W}}(\sigma \models \Box\Diamond A) = 1$ pour tout $t \in E^{\mathcal{F}}_{\Box\Diamond A}$.
2. Si π est une exécution fortement prob-équitable et équitable pour $\mathcal{F}$ telle que $\pi \models \Box\Diamond A$, alors $\mathtt{Inf}(\pi) \subseteq E^{\mathcal{F}}_{\Box\Diamond A}$.
3. Il existe un adversaire $\mathcal{W}$ à mémoire finie équitable pour $\mathcal{F}$ tel que $\Pr_{\mathcal{W}}(\sigma \models \Box A) = 1$ pour tout $\sigma \in E^{\mathcal{F}}_{\Box A}$.
4. Si π est une exécution fortement prob-équitable et équitable pour $\mathcal{F}$ telle que $\pi \models \Diamond\Box A$, alors $\mathtt{Inf}(\pi) \subseteq E^{\mathcal{F}}_{\Box A}$.

Preuve : On prouve 1. en suivant le même schéma que pour le Lemme 5.70 (cf. page 119).

Supposons π équitable pour $\mathcal{F}$, fortement prob-équitable et tel que $\pi \models \Box\Diamond A$. Alors il existe $F \subseteq \mathcal{F}$ tel que pour tout $f \in \mathcal{F} \setminus F$, f n'est tirable qu'un nombre fini de fois au cours de π, et pour tout $f \in F$, il existe une règle $\delta \in f$ tirée infiniment souvent le long de π. Montrons que l'ensemble des configurations visitées infiniment souvent au cours de π, c'est-à-dire $\mathtt{Inf}(\pi)$ est un post point fixe de $X \to (\mathsf{Conf} \setminus Pre[\mathcal{F} \setminus F](\mathsf{Conf})) \wedge \widehat{Pre}^+_X(A) \wedge \bigwedge_{f\in F} \widehat{Pre}^*_X(\widehat{Pre}_X[f](\mathsf{Conf}))$. Tout d'abord, si $\sigma \in \mathtt{Inf}(\pi)$, et $\delta \in \Delta(\sigma)$ est une règle tirée infiniment souvent au cours de π, alors $Post[\delta](\sigma) \subseteq \mathtt{Inf}(\pi)$. En ajoutant que $\pi \models \Box\Diamond A$, on obtient $\mathtt{Inf}(\pi) \subseteq Pre^+_{\mathtt{Inf}(\pi)}(A)$. Le caractère équitable pour $\mathcal{F}$ de π permet d'obtenir également que :

$$\mathtt{Inf}(\pi) \subseteq (\mathsf{Conf} \setminus Pre[\mathcal{F} \setminus F](\mathsf{Conf})) \wedge \bigwedge_{f\in F} \widehat{Pre}^*_{\mathtt{Inf}(\pi)}(\widehat{Pre}_{\mathtt{Inf}(\pi)}[f](\mathsf{Conf})).$$

Finalement, on obtient l'inclusion désirée ; $\mathtt{Inf}(\pi)$ est un post point fixe, et est donc inclus dans le plus grand point fixe, qui est $E^{\mathcal{F}}_{\Box\Diamond A}$.

La preuve de 4. est plus directe encore. Puisque $\pi \models \Diamond\Box A$, il existe un suffixe π' de π tel que $\pi' \models \Box A$. De plus, π' est équitable pour $\mathcal{F}$, fortement prob-équitable, et $\mathtt{Inf}(\pi') = \mathtt{Inf}(\pi)$. Le Lemme 5.70 (cf. page 119) implique alors que $\mathtt{Inf}(\pi) = \mathtt{Inf}(\pi') \subseteq I(\mathcal{F}, A)$. De $\pi \models \Diamond\Box A$ on tire également $\mathtt{Inf}(\pi) \subseteq A$. Alors $\mathtt{Inf}(\pi) \subseteq \widehat{Pre}^*_{\mathtt{Inf}(\pi)}(I(\mathcal{F}, A)) \wedge A$, en prenant un chemin vide pour la partie $\widehat{Pre}^*_{\mathtt{Inf}(\pi)}(I(\mathcal{F}, A))$. On en déduit que $\mathtt{Inf}(\pi)$ est inclus dans le plus grand point fixe de $X \to \widehat{Pre}^*_X(I(\mathcal{F}, A))$, soit $\mathtt{Inf}(\pi) \subseteq E^{\mathcal{F}}_{\Box A}$. □

Revenons à la preuve du Théorème 5.74.

Preuve de (a) : Soit $\sigma \in \nu X.\widehat{Pre}^*_X(E^{\mathcal{F}}_{\Box\Diamond A})$. Alors il existe un adversaire à mémoire finie $\mathcal{V}$ tel que $\Pr_{\mathcal{V}}(\sigma \models \Diamond E^{\mathcal{F}}_{\Box\Diamond A}) = 1$ (cf. Lemme 5.29, page 95). Soit alors $\mathcal{W}$ un adversaire à mémoire finie et équitable pour $\mathcal{F}$ tel que $\Pr_{\mathcal{W}}(\tau \models \Box\Diamond A) = 1$ pour toute configuration $\tau \in E^{\mathcal{F}}_{\Box\Diamond A}$. Il reste à combiner les adversaires $\mathcal{V}$ et $\mathcal{W}$ pour obtenir un adversaire $\mathcal{U}$ qui soit à mémoire finie, équitable pour $\mathcal{F}$ et vérifiant $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A) = 1$.

Réciproquement supposons que $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A) = 1$ pour un certain adversaire $\mathcal{U}$ à mémoire finie, et équitable pour $\mathcal{F}$. Alors, presque toutes les exécutions conformes à $\mathcal{U}$ sont fortement prob-équitables, équitables pour $\mathcal{F}$ et vérifient $\Box\Diamond A$. Soit π une telle exécution ;

$\texttt{Inf}(\pi) \subseteq E^{\mathcal{F}}_{\Box\Diamond A}$. On conclut que $\Pr_{\mathcal{U}}(\sigma \models \Diamond T^{\mathcal{F}}_{\Box\Diamond A}) = 1$. Finalement, à nouveau le Lemme 5.29 (cf. page 95) permet de déduire $\sigma \in \nu X.\widehat{Pre}^*_X(E^{\mathcal{F}}_{\Box\Diamond A})$.
Preuve de (b) : Soit $s \in Pre^*(E^{\mathcal{F}}_{\Box\Diamond A}))$. On choisit un chemin fini π partant de σ et atteignant $E^{\mathcal{F}}_{\Box\Diamond A}$. On suppose que π est simple (ne visite pas deux fois la même configuration) et que tout préfixe strict de π n'a aucune configuration dans $E^{\mathcal{F}}_{\Box\Diamond A}$:

$$\pi : \sigma \xrightarrow{*} \tau \in E^{\mathcal{F}}_{\Box\Diamond A}.$$

On définit un adversaire à mémoire finie de la façon suivante. L'adversaire $\mathcal{U}$ tente d'engendrer le chemin π. S'il échoue, il se comporte de façon arbitraire, mais équitable pour $\mathcal{F}$. Ceci est possible grâce à la Proposition 5.63 (cf. page 117). Si τ est atteint (c'est-à-dire si les pertes sont celles espérées) via le chemin π, alors $\mathcal{V}$ est un adversaire à mémoire finie et se comporte de façon équitable pour $\mathcal{F}$ en assurant $\Pr_{\mathcal{V}}(\tau \models \Box\Diamond A) = 1$ pour toute configuration $\tau \in E^{\mathcal{F}}_{\Box\Diamond A}$. L'adversaire combinant ces deux phases est à mémoire finie, équitable pour $\mathcal{F}$ et vérifie $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A) \geq \mathbb{P}(\pi) > 0$.

Réciproquement, soit $\mathcal{U}$ un adversaire à mémoire finie, équitable pour $\mathcal{F}$ avec $\Pr_{\mathcal{U}}(\sigma \models \Box\Diamond A) > 0$. On considère une exécution π conforme à $\mathcal{U}$, fortement prob-équitable et équitable pour $\mathcal{F}$ telle que $\pi \models \Box\Diamond A$. D'après le point 2., $\texttt{Inf}(\pi) \subseteq E^{\mathcal{F}}_{\Box\Diamond A}$. Et donc $\sigma \in Pre^*(E^{\mathcal{F}}_{\Box\Diamond A})$.
Preuve de (c) On montre plus précisément l'équivalence entre les assertions :

c.1 $\exists\mathcal{U}$ équitable pour $\mathcal{F}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Diamond\Box A) = 1$,

c.2 $\exists\mathcal{V}$ équitable pour $\mathcal{F}$ tel que $\Pr_{\mathcal{V}}(\sigma \models \Diamond E^{\mathcal{F}}_{\Box A}) = 1$,

c.3 $\exists\mathcal{W}$ tel que $\Pr_{\mathcal{W}}(\sigma \models \Diamond E^{\mathcal{F}}_{\Box A}) = 1$.

Preuve : L'équivalence entre $c.2$ et $c.3$ vient de la seconde partie du Lemme 5.64 (cf. page 117).

$c.1 \implies c.2$: Soit π une exécution infinie conforme à $\mathcal{U}$ partant de σ, fortement prob-équitable, équitable pour $\mathcal{F}$ et vérifiant $\pi \models \Diamond\Box A$. Par 4. on obtient $\texttt{Inf}(\pi) \subseteq E^{\mathcal{F}}_{\Box A}$. Puisque les exécutions équitables pour $\mathcal{F}$ et fortement prob-équitables vérifiant $\Diamond\Box A$ forment un ensemble de mesure 1, on obtient : $\Pr_{\mathcal{U}}(\sigma \models \Diamond E^{\mathcal{F}}_{\Box A}) = 1$.

$c.3 \implies c.1$: Supposons que $\mathcal{W}$ est un adversaire sans mémoire vérifiant $c.3$. De tels adversaires existent grâce au Lemme 5.29 (cf. page 95). On sait d'autre part, qu'il existe un adversaire $\mathcal{V}$ équitable pour $\mathcal{F}$ et à mémoire finie tel que $\Pr_{\mathcal{V}}(\tau \models \Box A) = 1$ pour toute configuration $\tau \in E^{\mathcal{F}}_{\Box A}$ (cf. 3.). En composant les adversaires $\mathcal{W}$ et V, on construit un adversaire $\mathcal{U}$ sans mémoire, équitable pour $\mathcal{F}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Diamond\Box A) = 1$. □

Preuve de (d) Si $\mathcal{U}$ est un adversaire à mémoire finie et équitable pour $\mathcal{F}$ tel que $\Pr_{\mathcal{U}}(\sigma \models \Diamond\Box A) > 0$, alors la mesure de l'ensemble des exécutions conformes à $\mathcal{U}$, équitables pour $\mathcal{F}$ et fortement prob-équitables telles que $\Diamond\Box A$ est vrai, est strictement positive. De plus pour chaque exécution de la sorte π, $\texttt{Inf}(\pi) \subseteq E^{\mathcal{F}}_{\Box A}$ grâce à 4. Ceci entraîne $\sigma \in Pre^*(E^{\mathcal{F}}_{\Box A})$.

Vice et versa, soit $\sigma \in Pre^*(E^{\mathcal{F}}_{\Box A})$. Il existe un chemin simple $\pi : \sigma \xrightarrow{*} \tau \in E^{\mathcal{F}}_{\Box A}$. On considère alors l'adversaire $\mathcal{U}$ à mémoire finie et équitable pour $\mathcal{F}$ qui cherche à réaliser π. S'il échoue, il se comporte de façon arbitraire mais équitable pour $\mathcal{F}$, sinon, il adopte à partir de τ le comportement d'un adversaire qui permet de vérifier $\Box A$ avec probabilité 1. C'est possible car $\tau \in E^{\mathcal{F}}_{\Box A}$. Ainsi $\Pr_{\mathcal{U}}(\sigma \models \Diamond\Box A) > 0$. □

Corollaire 5.75. *Le problème, étant donné $\mathcal{N}$ un NPLCS, $\mathcal{F} \subseteq 2^{\Delta}$ une contrainte d'équité, $R \subseteq \mathsf{Conf}$ une* région *et $\sigma \in \mathsf{Conf}$ une configuration initiale, de savoir s'il existe un adversaire $\mathcal{U}$ à mémoire finie et équitable pour $\mathcal{F}$ tel que* (*a*) *(resp.* (*b*), (*c*) *ou* (*d*)*) est décidable.*

(a) $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond R) = 1$,

(b) $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond R) > 0$,

(c) $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond R) < 1$,

(d) $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond R) = 0$.

Preuve : Le Théorème 5.74 (cf. page 121) donne des caractérisations des ensembles de configurations à partir desquelles il existe un adversaire équitable vérifiant les bonnes propriétés. Pour prouver le Corollaire 5.75, il suffit de montrer que les termes définissant ces régions sont gardés, et d'appliquer le Théorème 1.26 (cf. page 32).

Si R est une région, alors $E^{\mathcal{F}}_{\Box\Diamond R}$ qui s'écrit comme le terme gardé :

$$\nu X.\Big(\mathsf{Conf} \setminus \big(\mathit{Pre}[\mathcal{F} \setminus F](\mathsf{Conf})\big)\Big) \wedge \widehat{\mathit{Pre}}^{+}_{K_{\downarrow}X}(R) \wedge \bigwedge_{f \in F} \widehat{\mathit{Pre}}^{*}_{K_{\downarrow}X}\big(\widehat{\mathit{Pre}}_{K_{\downarrow}X}[f](\mathsf{Conf})\big)$$

grâce à la Proposition 5.22 (cf. page 93), est effectivement calculable.

On en déduit la calculabilité de $E^{\mathcal{F}}_{\Box\Diamond R}$, qui est l'union des $E^{\mathcal{F}}_{\Box\Diamond R}$. Cela permet de prouver la décidabilité pour (b) et (d).

Pour prouver que les termes pour (a) et (c) sont gardés, on utilise les résultats obtenus jusqu'ici et à nouveau la Proposition 5.22 (cf. page 93) :

$$\nu X.\widehat{\mathit{Pre}}^{*}_{X}(E^{\mathcal{F}}_{\Box\Diamond R}) = \nu X.\widehat{\mathit{Pre}}^{*}_{K_{\downarrow}X}(E^{\mathcal{F}}_{\Box\Diamond R}), \tag{a}$$

$$\nu X.\widehat{\mathit{Pre}}^{*}_{X}(E^{\mathcal{F}}_{\Box R}) = \nu X.\widehat{\mathit{Pre}}^{*}_{K_{\downarrow}X}(E^{\mathcal{F}}_{\Box R}). \tag{c}$$

Ceci termine la preuve du Corollaire 5.75. □

Dans ce chapitre nous avons prouvé l'indécidabilité de la vérification qualitative de propriétés exprimées par des formules du temps linéaire pour les NPLCS. En restreignant le pouvoir des adversaires (plus précisément en considérant des adversaires à mémoire finie), nous avons montré que le même problème était décidable. Enfin, nous avons étendu ce résultat à des adversaires comportant une hypothèse d'équité. Dans la partie suivante, nous expliquons comment nous avons exploité ces résultats pour implémenter un outil de vérification des NPLCS.

Quatrième partie

Mise en œuvre des techniques de vérification

Chapitre 6

Implémentation et études de cas

Sommaire

À partir des algorithmes présentés à la fois dans le Chapitre 1 et le Chapitre 5 nous avons implémenté un prototype de model-checker, programmé en Caml, pour la vérification qualitative de NPLCS. C'est ce travail (publié en partie dans [BBS06c]) que nous décrivons dans ce chapitre. Nous commençons par donner des détails sur le type de régions que nous avons considéré (une sous classe suffisamment expressive des régions régulières) et leurs algorithmes spécifiques. Dans un deuxième temps, nous exposons deux études de cas que nous avons menées grâce à notre outil : nous montrons sous certaines hypothèses d'équité une propriété de vivacité et de progrès dans le protocole du Bit Alterné, puis nous prouvons pour le protocole de Pachl [Pac87] à la fois des propriétés de sûreté, et des propriétés de vivacité pour des adversaires équitables. Ces études couvrent donc la plupart des techniques présentées dans le Chapitre 5 : propriété de sûreté, de vivacité, et pour des adversaires équitables ou généraux.

6.1 Implémentation des algorithmes symboliques

6.1.1 Régions et algorithmes

Le model-checking symbolique est basé sur des objets symboliques qui représentent des ensembles de configurations, et des méthodes algorithmiques pour manipuler ces objets. Dans cette section, nous présentons un cadre symbolique pour les NPLCS, basé sur les *différences de clôtures vers le haut avec préfixe*. Cette représentation étend des techniques précédentes proposées dans [ACBJ04, KS06] puisqu'elle permet de traiter la différence d'ensembles, et de vérifier quel est le message en tête de chaque canal. Par rapport aux langages réguliers elle est moins expressive (mais suffisamment expressive pour notre étude) mais plus facile à manipuler sous forme de termes Caml. Par souci de simplification, nous supposons dans la présentation que les NPLCS considérés ne possèdent qu'un canal de communication. Les algorithmes implémentés ne font pas cette hypothèse comme en témoigneront les études de cas.

Rappelons qu'un ensemble $T \subseteq \mathsf{Conf}$ est *fermé vers le haut* (resp. *fermé vers le bas*) si pour toute configuration $\sigma \in T$, et pour tout $\tau \sqsupseteq \sigma$ (resp. $\tau \sqsubseteq \sigma$), $\tau \in T$. Aussi, pour $T \subseteq \mathsf{Conf}$, $\uparrow T$ dénote la fermeture vers le haut[1] de T (c'est-à-dire le plus petit ensemble fermé vers le haut contenant T), et $\downarrow T$ dénote la fermeture vers le bas de T (plus petit fermé vers le bas contenant T). Pour les singletons on notera $\uparrow\sigma$ (resp. $\downarrow \sigma$) pour $\uparrow\{\sigma\}$ (resp. $\downarrow \{\sigma\}$).

Les ensembles symboliques que nous considérons sont termes définis par la grammaire suivante :

$$\begin{array}{rrcll}
\text{préfixe :} & \alpha & := & \varepsilon \mid m & m \in \mathsf{M} \\
\text{clôture préfixée :} & \theta & := & \alpha{\uparrow}u & u \in \mathsf{M}^* \\
\text{somme de clôtures préfixées :} & \sigma & := & \theta_1 + \cdots + \theta_n & n \geq 0 \\
\text{ensemble symbolique simple :} & \rho & := & \langle s, \theta - \sigma \rangle & s \in S \text{ état de contrôle} \\
\text{ensemble symbolique :} & \gamma & := & \rho_1 + \cdots + \rho_n & n \geq 0
\end{array}$$

Les clôtures (vers le haut) préfixées et leurs sommes représentent des sous-ensembles de M^* dont la sémantique est précisée ci-dessous :

$$\llbracket \alpha{\uparrow}u \rrbracket \stackrel{\text{def}}{=} \{\alpha v \mid u \sqsubseteq v\},$$
$$\llbracket \theta_1 + \cdots + \theta_n \rrbracket \stackrel{\text{def}}{=} \llbracket \theta_1 \rrbracket \cup \cdots \cup \llbracket \theta_n \rrbracket.$$

Les ensembles symboliques représentent eux des sous-ensembles de l'ensemble des configurations Conf définis par

$$\llbracket \langle q, \theta - (\theta_1 + \cdots + \theta_n) \rangle \rrbracket \stackrel{\text{def}}{=} \{\langle q, v \rangle \in \mathsf{Conf} \mid v \in \llbracket \theta \rrbracket \smallsetminus (\llbracket \theta_1 \rrbracket \cup \cdots \cup \llbracket \theta_n \rrbracket)\}$$

Dans la première partie (cf. partie I), nous avons introduit les *régions régulières* qui sont des ensembles de configurations dont les contenus de canaux forment des langages réguliers. Dans ce chapitre, on appellera région les ensembles de configurations qui correspondent à la sémantique des ensembles symboliques.

Définition 6.1 (Région)**.** *On appelle* région *tout sous-ensemble $A \subseteq \mathsf{Conf}$ qui peut être dénoté par un ensemble symbolique : $\exists\gamma$ tel que $A = \llbracket\gamma\rrbracket$.*

Une région de contrôle *est une région qui peut être dénotée par $\sum_i \langle s_i, \varepsilon{\uparrow}\varepsilon \rangle$, c'est-à-dire où les contenus des canaux ne sont pas contraints.*

Par abus de notation, $\emptyset$ dénotera à la fois la somme vide (*i.e.* $n = 0$) de clôtures préfixées et l'ensemble symbolique vide. Parfois, on écrit $\uparrow v$ pour $\varepsilon{\uparrow}v$, c'est-à-dire si le préfixe est ε, et $\theta - \theta_1 - \cdots - \theta_n$ pour $\theta - (\theta_1 + \cdots + \theta_n)$, et θ pour $\theta - \emptyset$. Enfin, on écrit $\gamma \equiv \gamma'$ quand $\llbracket\gamma\rrbracket = \llbracket\gamma'\rrbracket$, *i.e.* quand γ et γ' représentent la même région.

Lemme 6.2. *L'appartenance d'une configuration à une région est décidable.*

Théorème 6.3 (Calcul symbolique : les bases)**.**

[1] Notée $C_{\uparrow}T$ dans la Partie I.

Fermeture booléenne : *Les régions sont closes par union, intersection et complément. De plus, il existe des algorithmes qui, étant donnés des ensembles symboliques γ_1 et γ_2 renvoient des ensembles symboliques (notés $\gamma_1 \sqcup \gamma_2$, $\gamma_1 \sqcap \gamma_2$ et $\neg\gamma$) tels que $[\![\gamma_1 \sqcup \gamma_2]\!] = [\![\gamma_1]\!] \cup [\![\gamma_2]\!]$, $[\![\gamma_1 \sqcap \gamma_2]\!] = [\![\gamma_1]\!] \cap [\![\gamma_2]\!]$ et $[\![\neg\gamma]\!] = \mathsf{Conf} \smallsetminus [\![\gamma]\!]$.*

Fermeture vers le haut : *Les régions sont closes par fermeture vers le haut. De plus, il existe un algorithme qui, étant donné un ensemble symbolique γ renvoie un ensemble symbolique noté $\uparrow\gamma$ tel que $[\![\uparrow\gamma]\!] = \uparrow[\![\gamma]\!]$.*

Vide : *On peut décider si $[\![\gamma]\!] = \emptyset$ étant donné un ensemble symbolique γ.*

Prédécesseurs : *Les régions sont closes pour les opérateurs $Pre(\_)$ et $\widehat{Pre}_{\_}(\_)$. De plus, il existe des algorithmes qui, étant donnés des ensembles symboliques γ et γ' renvoient des ensembles symboliques notés $Pre(\gamma)$ et $\widehat{Pre}_{\gamma'}(\gamma)$ tels que $[\![Pre(\gamma)]\!] = Pre([\![\gamma]\!])$ et $[\![\widehat{Pre}_{\gamma'}(\gamma)]\!] = \widehat{Pre}_{[\![\gamma']\!]}([\![\gamma]\!])$.*

Preuve : <u>Union :</u>
On peut prendre $\gamma_1 + \gamma_2$ pour $\gamma_1 \sqcup \gamma_2$.

<u>Intersection :</u>
On commence par considérer des termes simples. Pour deux clôtures $\uparrow u$ et $\uparrow v$, l'intersection $\uparrow u \sqcap \uparrow v$ est la somme $\tau = \sum_{i=1}^{n} \uparrow w_i$ où les w_i sont les mots minimaux de $[\![\uparrow u]\!] \cap [\![\uparrow v]\!]$. Par exemple, $\uparrow 010 \sqcap \uparrow 210 = \uparrow 2010 + \uparrow 01201 + \uparrow 02101$. Les w_i peuvent être calculés de façon effective puisque la longueur de chacun des w_i est bornée par $|u|+|v|$. Par ailleurs, le nombre d'éléments minimaux et borné :

$$n \leq \binom{|u|+|v|}{|u|} = \binom{|u|+|v|}{|v|}.$$

Cette borne est atteinte en particulier lorsque u et v ne partagent aucun message. L'algorithme pour l'intersection n'est donc pas polynomial.

Pour les clôtures préfixées, on introduit les opérateurs qui donnent le résiduel (gauche ou droit) d'un mot par un message. Soit $u \in \mathsf{M}^*$ et $m \in \mathsf{M}$. On définit :

$$m^{-1}u \stackrel{\text{def}}{=} \begin{cases} v & \text{si } u = mv, \\ u & \text{sinon,} \end{cases} \qquad u\, m^{-1} \stackrel{\text{def}}{=} \begin{cases} v & \text{si } u = vm, \\ u & \text{sinon.} \end{cases}$$

On peut alors réduire l'intersection des clôtures préfixées aux cas précédents (clôtures) par :

$$\alpha{\uparrow}u \sqcap \beta{\uparrow}v \stackrel{\text{def}}{=} \begin{cases} \alpha(\uparrow u \sqcap \uparrow v) & \text{si } \alpha = \beta, \\ \alpha(\uparrow u \sqcap \uparrow(\alpha^{-1}v)) & \text{si } \alpha \in \mathsf{M} \text{ et } \beta = \varepsilon, \\ \beta(\uparrow(\beta^{-1}u) \sqcap \uparrow v) & \text{si } \beta \in \mathsf{M} \text{ et } \alpha = \varepsilon, \\ \emptyset & \text{si } \alpha \neq \beta, \text{ et } \alpha, \beta \in \mathsf{M}, \end{cases}$$

où $\alpha(\uparrow u \sqcap \uparrow v)$ est un raccourci pour $\sum_i \alpha{\uparrow}w_i$ quand $\uparrow u \sqcap \uparrow v = \sum_i \uparrow w_i$.

Enfin pour l'intersection des ensembles symboliques simples et des ensembles symboliques, il suffit d'utiliser :

$$\langle s, \theta - \sigma \rangle \sqcap \langle s', \theta' - \sigma' \rangle \stackrel{\text{def}}{=} \begin{cases} \sum_i \langle s, \theta_i - \sigma - \sigma' \rangle & \text{si } s = s' \text{ et } \theta \sqcap \theta' = \sum_i \theta_i, \\ \emptyset & \text{sinon,} \end{cases}$$

$$\sum_i \rho_i \prod \sum_j \rho_j \stackrel{\text{def}}{=} \sum_{i,j} (\rho_i \sqcap \rho_j).$$

Complément :
Pour les ensembles symboliques simples :

$$\neg\big\langle s, \alpha{\uparrow}v - \sum_{i=1}^{n} \beta_i{\uparrow}w_i\big\rangle \stackrel{\text{def}}{=} \langle s, {\uparrow}\varepsilon - \alpha{\uparrow}v\rangle + \sum_{i=1}^{n}\langle s, \beta_i{\uparrow}w_i\rangle + \sum_{s'\neq s}\langle s, {\uparrow}\varepsilon\rangle.$$

En utilisant l'intersection, ceci permet de calculer le complément de tout ensemble symbolique, puisque

$$\neg\,\big(\sum_i \rho_i\big) \stackrel{\text{def}}{=} \prod_i \neg\rho_i.$$

Fermeture vers le haut :
Le cas difficile est la fermeture vers le haut d'une différence $\alpha{\uparrow}u - \sigma$. Clairement, si $\alpha\, u \notin [\![\sigma]\!]$, alors on peut poser ${\uparrow}(\alpha{\uparrow}u - \sigma) \stackrel{\text{def}}{=} {\uparrow}(\alpha\, u)$. Néanmoins, il est possible d'avoir simultanément $\alpha\, u \in [\![\sigma]\!]$ et $\alpha{\uparrow}u - \sigma \not\equiv \emptyset$. Par exemple ${\uparrow}ab - a{\uparrow}b \equiv b{\uparrow}ab$ quand $\mathsf{M} = \{a, b\}$.

Cette difficulté repose sur la possibilité à la fois de spécifier le message de tête du canal mais aussi sur la présence de restrictions (l'ensemble σ). Dans les travaux existants, l'un de ces attributs n'était pas présent. Par exemple, dans le cadre plus simple des *ensembles contraints* de [KS06], les ensembles considérés ne sont pas préfixés, mais il y a comme ici un ensemble des restrictions. Les *contraintes* introduites dans [ABd03] ne permettent que de considérer des sommes d'ensembles fermés vers le haut préfixées (d'ailleurs avec pour préfixe un mot, plutôt qu'une lettre comme ici), mais pas de différences. Les ensembles symboliques que nous avons introduits combinent ces deux aspects, ce qui complexifie les algorithmes.

Pour la fermeture vers le haut, la solution générale s'exprime ainsi :

$${\uparrow}(\alpha{\uparrow}u - \sigma) \stackrel{\text{def}}{=} \begin{cases} {\uparrow}(\alpha u) & \text{si } \alpha u \notin [\![\sigma]\!], \\ \emptyset & \text{si } (\alpha \in \mathsf{M} \text{ ou } \alpha = u = \varepsilon) \text{ et } \alpha u \in [\![\sigma]\!], \\ \sum_{m'\neq m} {\uparrow}(m'{\uparrow}u - \sigma) & \text{si } \alpha = \varepsilon,\ u = mv, \text{ et } u \in [\![\sigma]\!]. \end{cases}$$

Le troisième cas est traité de façon récursive en utilisant les deux premiers, mais la définition reste bien fondée.

Vide :
Le vide est décidable pour les ensembles γ sans restrictions, c'est-à-dire les sommes de clôtures préfixées. Pour les ensembles avec restrictions, on se ramène au cas simple (sans restriction) grâce à $\gamma \equiv \emptyset$ ssi ${\uparrow}\gamma \equiv \emptyset$.

Prédécesseurs :
Commençons par $\mathit{Pre}[\delta]\langle s, {\uparrow}v\rangle$. Il faut distinguer trois cas, selon le type de δ :

$$\mathit{Pre}[t \xrightarrow{c?m} s]\langle s, {\uparrow}v\rangle \stackrel{\text{def}}{=} \langle t, m{\uparrow}v\rangle, \qquad \text{(Lecture)}$$

$$\mathit{Pre}[t \xrightarrow{c?m} s]\langle s, {\uparrow}v\rangle \stackrel{\text{def}}{=} \langle t, {\uparrow}(v\, m^{-1})\rangle, \qquad \text{(Écriture)}$$

$$\mathit{Pre}[t \xrightarrow{\surd} s]\langle s, {\uparrow}v\rangle \stackrel{\text{def}}{=} \langle t, {\uparrow}v\rangle, \qquad \text{(Action interne)}$$

alors que $\mathit{Pre}[t \xrightarrow{op} s']\langle s, {\uparrow}v\rangle = \emptyset$ si $s' \neq s$.

Ceci permet de calculer les prédécesseurs en une étape d'ensembles de configurations arbitraires puisque $\mathit{Pre}[\delta](\bigcup S_i) = \bigcup_i \mathit{Pre}[\delta](S_i)$ et, $\mathit{Pre}[\delta](S) = \mathit{Pre}[\delta]({\uparrow}S)$ (cf. Lemme 1.21, page 30).

Les algorithmes sont donnés par :

$$\begin{aligned} Pre(\gamma) &= \bigcup_{\delta\in\Delta} Pre[\delta](\gamma), \\ Pre(\rho_1 + \cdots + \rho_n) &= Pre(\rho_1) + \cdots + Pre(\rho_n), \\ Pre(\langle s, \theta - \sigma\rangle) &= Pre(\langle s, \uparrow(\theta - \sigma)\rangle). \end{aligned}$$

Ainsi, $Pre[\delta](\gamma)$ est une somme de clôtures préfixées (dans le cas d'une lecture) ou une somme de clôtures (dans le cas d'une écriture ou d'une action interne).

Prédécesseurs contraints :
Rappelons la définition de l'opérateur $\widehat{Pre}$.

$$\widehat{Pre}_{\gamma'}(\gamma) \stackrel{\text{def}}{=} \bigsqcup_{\delta\in\Delta} \big(Pre[\delta](\gamma) \setminus Pre[\delta](\neg\gamma')\big).$$

Puisque Pre peut être calculé, il est en de même de $\widehat{Pre}$. □

Le Théorème 6.3 (cf. page 128) fournit les éléments de base nécessaires au model-checking symbolique des LCS. Ces ingrédients peuvent alors être utilisés pour calculer des ensembles définis comme points fixes. Par exemple, un ensemble symbolique représentant $Pre^*(\llbracket\gamma\rrbracket)$ est défini par $\mu X.\gamma \sqcup Pre(X)$ que l'on peut tenter de calculer itérativement. Dans la Partie I de cette thèse nous avons montré que ce calcul itératif termine dans le cadre de l'algèbre des régions régulières. Ce résultat est encore applicable ici, pour une nouvelle algèbre des régions où les régions sont les ensembles symboliques.

Plus précisément, l'algèbre des régions que nous considérons a pour famille d'opérateurs monotones

$$O = \{\sqcup, \sqcap, C_\uparrow, C_\downarrow, K_\uparrow, K_\downarrow, Pre, \widetilde{Pre}, \widehat{Pre}\}$$

et pour régions les ensembles symboliques.

Le théorème précédent permet d'affirmer que les ensembles symboliques sont stables par les opérateurs de O. Toujours grâce au Théorème 6.3 cette algèbre des régions est effective. On peut donc appliquer le Théorème 1.26 (cf. page 32) dans ce cadre restreint des régions régulières.

On rappelle que si ϕ dénote une formule du temps linéaire, $\mathbf{E}[\phi]$ dénote l'ensemble des configurations σ pour lesquelles il existe un adversaire $\mathcal{U}$ tel que $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \phi) = 1$.

Nous pouvons à présent énoncer un théorème concernant certains calculs de points fixes dans l'algèbre des régions des ensembles symboliques.

Théorème 6.4 (Calcul symbolique : points fixes)**.**

Prédécesseurs (contraints) itérés : *Les régions sont stables par les opérateurs* $Pre^*(\_)$ *et* $\widehat{Pre}^*_{\_}(\_)$. *De plus, il existe des algorithmes qui, étant donnés des ensembles symboliques* γ *et* γ' *renvoient des termes notés* $Pre^*(\gamma)$ *et* $\widehat{Pre}^*_{\gamma'}(\gamma)$, *tels que* $\llbracket Pre^*(\gamma)\rrbracket = Pre^*(\llbracket\gamma\rrbracket)$ *et* $\llbracket\widehat{Pre}^*_{\gamma'}(\gamma)\rrbracket = \widehat{Pre}^*_{\llbracket\gamma'\rrbracket}(\llbracket\gamma\rrbracket)$.

$\mathbf{E}[\Box\gamma]$ *Pour toute région* γ, *l'ensemble* $\mathbf{E}[\Box\gamma]$ *est une région, pour laquelle on peut calculer un terme.*

$\mathbf{E}[\Diamond\gamma]$ *Pour toute région* γ, *l'ensemble* $\mathbf{E}[\Diamond\gamma]$ *est une région, pour laquelle on peut calculer un terme.*

∃CTL : *L'ensemble des configurations satisfaisant une formule ∃CTL (*i.e.* une formule de CTL où seules les modalités «∃(_ Until _)» et «∃ X _» sont permises) est une région quand les propositions atomiques sont elles-mêmes des régions. De plus, on peut calculer un ensemble symbolique pour cette région à partir de la formule ∃CTL.*

Preuve : On a déjà donné pour ces ensembles des termes de L_μ que l'on rappelle ci-dessous.

$$\begin{aligned} Pre^*(\gamma) &= \mu X.\big(\gamma \sqcup Pre(C_\uparrow X)\big), & \widehat{Pre}^*_\gamma(\gamma') &= \mu X.\big(\gamma' \sqcup \widehat{Pre}_\gamma(C_\uparrow X)\big), \\ \mathbf{E}[\Box\gamma] &= \nu X.\big(\gamma \sqcap \widehat{Pre}_{K_\downarrow X}(\mathsf{Conf})\big), & \mathbf{E}[\Diamond\gamma] &= \nu X.\widehat{Pre}^*_{K_\downarrow X}(\gamma), \qquad (*) \\ \exists\big(\gamma \textit{ Until } \gamma'\big) &= \mu X.\Big(\gamma' \sqcup \big(\gamma \sqcap Pre(C_\uparrow X)\big)\Big), & \exists\big(\mathsf{X}\,\gamma\big) &= Pre(\gamma). \end{aligned}$$

Tous ces termes sont gardés, et l'application du Théorème 1.26, sur la convergence du calcul itératif des termes gardés, permet de conclure. □

6.1.2 Implémentation

Nous avons réalisé une implémentation en Caml du calcul symbolique sur les ensembles de configurations présenté ci-dessus.

Nous avons défini des structures de données simples pour les ensembles symboliques, et les opérateurs introduits plus haut se codent eux aussi facilement grâce aux algorithmes donnés dans la preuve du Théorème 6.3 (cf. page 128). Le point le plus délicat a été de pouvoir tester l'égalité de deux ensembles symboliques.

Nous introduisons une notion de *forme réduite*, à chaque niveau de termes : c'est-à-dire aussi bien pour les différences de clôtures préfixées que pour les ensembles symboliques simples. Avant de définir ces formes réduites, faisons le tour des structures dont nous avons besoin. Ici encore on simplifie la présentation en supposant qu'il n'y a qu'un canal de communication.

Le premier type d'ensembles que nous considérons, nommé `uclos` (pour union de clos) permettra de représenter les ensembles fermés vers le haut, c'est à dire de la forme $\uparrow\sigma_1+\cdots+\uparrow\sigma_n$. Un ensemble dénoté par une seule configuration sera du sous type `clos`. On définit également des type `preclos` et `upclos` pour les clôtures préfixées et les unions de telles structures. Les ensembles symboliques simples sont créés à partir d'un état de contrôle, un `preclos` et d'un `upclos`.

Définition 6.5. *Un ensemble de type* `uclos` *est dit en* forme réduite *s'il est représenté par ses éléments minimaux.*

Ainsi, deux `uclos` en forme réduite représentent le même ensemble, si et seulement si, ils ont le même ensemble de représentants.

Étant données deux clôtures préfixées $\theta = \alpha{\uparrow}u$ et $\theta' = \alpha'{\uparrow}u'$, on dit que θ et θ' sont incomparables, si les ensembles qu'elles dénotent le sont : $[\![\theta]\!] \not\subseteq [\![\theta']\!]$, et $[\![\theta']\!] \not\subseteq [\![\theta]\!]$.

Proposition 6.6. *On peut décider si deux clôtures préfixées sont incomparables.*

Preuve : Soient $\theta = \alpha{\uparrow}u$ et $\theta' = \alpha'{\uparrow}u'$. Alors,

$$[\![\alpha{\uparrow}v]\!] \subseteq [\![\alpha'{\uparrow}v']\!] \text{ ssi } \begin{cases} \alpha = \alpha' \text{ et } v \sqsubseteq v' & \text{ou,} \\ \alpha \in \mathsf{M},\ \alpha' = \varepsilon \text{ et } \alpha v \sqsubseteq v'. \end{cases}$$

Cette caractérisation permet de déduire la décidabilité de $[\![\theta]\!] \subseteq [\![\theta']\!]$. □

Définition 6.7. *Un* `upclos` *est en* forme réduite *si toutes les clôtures préfixées qui le composent sont incomparables deux à deux.*

Un ensemble symbolique est représenté par l'ensemble des différences ρ_i qui le composent : $\gamma = \rho_1 + \cdots + \rho_n$.

Définition 6.8. *Un ensemble symbolique est en* forme réduite *si* $\gamma = \emptyset$ *ou* $\gamma = \rho_1 + \cdots + \rho_n$ *avec* $n \geq 1$ *et pour tout indice* $1 \leq i \leq n$, $\rho_i = \langle q_i, \theta_i - \sigma_i \rangle$ *avec :*
- *soit* $\theta_i = \uparrow\varepsilon$, *soit* $\theta_i = m\uparrow u$ *avec* $m \in \mathsf{M}$ *et* $u \in \mathsf{M}^*$,
- σ_i *est un* `upclos` *en forme réduite,*
- $\theta_i \not\sqsubseteq \sigma_i$.

Procédure – Mise sous forme réduite des ensembles symboliques

Entrée Un ensemble symbolique $\gamma = \rho_1 + \cdots + \rho_n$.
Sortie Un ensemble symbolique γ' en forme réduite et tel que $[\![\gamma']\!] = [\![\gamma]\!]$.
début
pour tout ensemble symbolique simple $\rho_i = \langle s_i, \theta_i - \sigma_i \rangle$,
1. **si** $[\![\theta_i]\!] \subseteq [\![\sigma_i]\!]$, remplacer ρ_i par $\emptyset$.
2. **sinon**, mettre σ_i en forme réduite
3a. **si** $\theta_i = \uparrow\varepsilon$ ou $\theta_i = m\uparrow v$ avec $m \in \mathsf{M}$, laisser θ_i inchangé
3b. **sinon**, $\theta_i = \varepsilon\uparrow v$, récrire θ_i sous la forme $\sum_{m \in \mathsf{M}} m\uparrow(m^{-1}v)$; puis distribuer les restrictions σ_i et effectuer les simplifications nécessaires.[a]
fin

[a]Des simplifications peuvent apparaître si $[\![m\uparrow(m^{-1}v)]\!] \subseteq [\![\sigma_i]\!]$. Dans ce cas, à l'étape 3b on obtient par exemple un terme de la forme $(a\uparrow w + b\uparrow v) - a\uparrow w - a\uparrow w'\uparrow w''$, et on le simplifie en $b\uparrow v - \uparrow w''$.

En plus de faciliter le test d'égalité, les formes réduites permettent de calculer la clôture vers le haut d'un ensemble symbolique. En effet, lorsqu'un ensemble symbolique est sous forme réduite, pour tout ρ_i, l'élément minimal de θ_i est aussi l'élément minimal de ρ_i, ce qui permet de prendre facilement la fermeture vers le haut d'un ensemble symbolique.

À partir des fonctions de base que nous avons implémentées (union, intersection, négation, Pre, $\widehat{Pre}$, $\widetilde{Pre}$, etc...), nous pouvons définir des procédures calculant l'ensemble des configurations pour lesquelles il existe un adversaire satisfaisant une formule avec probabilité 1 ou positive. Pour cela on utilise les définitions de ces ensembles par points fixes, fournis dans le Chapitre 5.

Une fois ces procédures implémentées, nous les avons testées sur deux principaux protocoles, le protocole du Bit Alterné, et une variante plus complexe, le protocole de Pachl. Notre outil a permis d'obtenir en particulier des preuves de la vivacité, et du bon fonctionnement de ces protocoles sous certaines hypothèse d'équité. Nous détaillons ces deux études de cas dans la suite.

6.2 Vérification automatique du protocole du bit alterné

Le protocole du bit alterné a été introduit par Bartlett *et al.* [BSW69]. C'est un protocole de communication entre deux composants qui peut être utilisé lorsque les canaux de communications sont non fiables. Depuis son introduction, il a été étudié maintes fois, mais jamais à notre connaissance des propriétés de vivacité n'ont pu être prouvées automatiquement (sans faire d'hypothèses supplémentaires – canaux finis par exemple). Une illustration de ce proto-

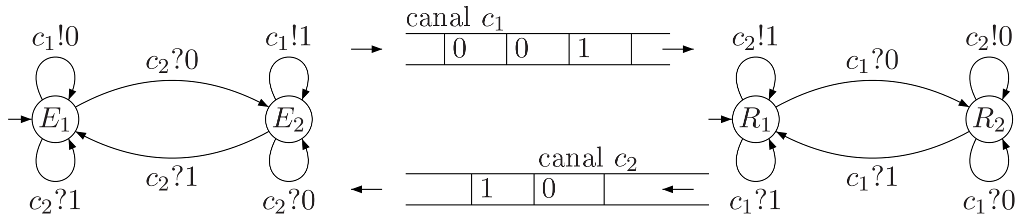

FIG. 6.1 – Une modélisation du protocole du Bit Alterné

cole est donnée à la Figure 6.1 (qui reprend le schéma de la Figure 1.2, page 20). Les états de contrôle de l'émetteur (resp. du récepteur) sont E_1 et E_2 (resp. R_1 et R_2).

Décrivons le comportement «normal» (c'est-à-dire lorsqu'il n'y a pas de perte de messages) du protocole du bit alterné. Les canaux c_1 et c_2 ont des fonctions différentes et chaque composant peut lire dans un canal et écrire dans l'autre : l'émetteur envoie ses messages sur le canal c_1 et lit les accusés de réception que le récepteur envoie sur le canal c_2. La configuration initiale est $\texttt{Init} = (E_1, R_1, \varepsilon, \varepsilon)$: l'émetteur et le récepteur se trouvent tous les deux dans leur état initial, et les canaux sont vides. L'émetteur peut alors envoyer un message sur le canal c_1. On fait abstraction du contenu des messages dans cette modélisation ; on ne garde que le bit de contrôle des messages qui alterne entre 0 et 1 (d'où le nom de *protocole du bit alterné*). La première action effectuée par l'émetteur est donc $c_1!0$ qui écrit le message étiqueté 0 dans le canal c_1. L'automate reste dans l'état de contrôle E_1R_1. Le récepteur peut alors lire le message envoyé et faire passer le système à l'état E_1R_2. Les rôles sont alors inversés, car le récepteur, pour accuser la réception du message 0, va envoyer un message étiqueté également 0 sur le canal c_2. Quand l'émetteur reçoit ce message, il peut passer à l'envoi de son second message qui sera étiqueté par le bit 1. Ce comportement idéal ne prend pas en compte les pertes possibles des messages lorsqu'ils sont en transit. C'est pourquoi, dans le protocole du Bit Alterné, les émissions de messages sont des boucles d'émission permettant une nouvelle émission en cas de pertes du message ou de l'accusé de réception. En pratique, des délais d'attente sont déclenchés lorsqu'un message est émis et, si aucun accusé correspondant n'est reçu avant la fin de cette attente, le message est émis à nouveau. Nous avons abstrait ces délais d'attente dans notre modélisation. La Figure 6.2 présente le produit asynchrone des deux composants du protocole du Bit Alterné. Pour une meilleure lisibilité, nous avons représenté sur une seule boucle l'ensemble des actions qui permettent de boucler sur un état de contrôle. De plus, par rapport aux composants de la Figure 6.1, nous avons supprimé certaines règles de transitions : les boucles de lecture. En effet, ces transitions sont inutiles grâce à la sémantique que nous avons donnée aux pertes[2].

Le fonctionnement espéré du protocole du bit alterné comporte deux aspects : Tout d'abord, il ne faut pas de confusion entre les messages. C'est le rôle du bit de contrôle, et cette propriété de sûreté a déjà été prouvée dans la littérature. D'autre part, on attend qu'il y ait effectivement une communication entre les deux composants, ce qui revient à dire que le système visite chacun de ses états de contrôle infiniment souvent.

Cette deuxième propriété nécessite de faire des hypothèses d'équité sur les exécutions. En effet, si par exemple aucune hypothèse n'est faite sur les pertes de messages, on peut imaginer

[2]Rappelons que ces boucles de lecture étaient essentielles dans le modèle *Front Lossy* de Finkel [Fin94]

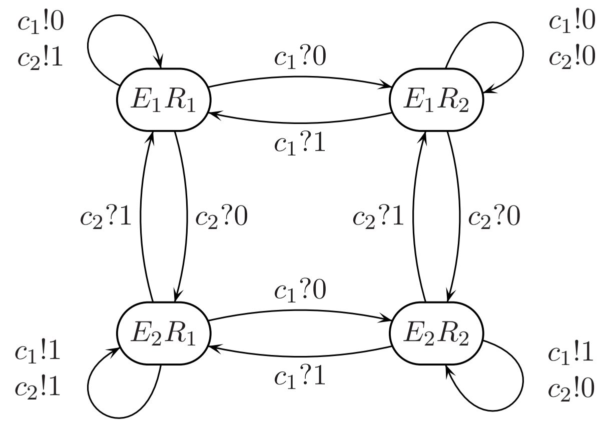

FIG. 6.2 – Le produit des composants du Bit Alterné

un scénario où tous les messages sont perdus, et donc aucune communication n'est possible. On voit par cet exemple la nécessité d'ajouter une hypothèse d'équité sur les pertes de messages : par exemple «si un message est envoyé infiniment souvent, il doit finir par être transmis». La sémantique probabiliste que nous donnons aux pertes dans les NPLCS entraîne cette équité sur les pertes.

Cependant, il faut faire d'autres hypothèses d'équité pour garantir le fonctionnement correct du Bit Alterné. Il s'agit dans un premier temps d'instaurer une équité entre les composants. En effet, si un des composants (émetteur ou récepteur) est le seul à partir d'un moment à exécuter des actions, le système restera bloqué dans un état de contrôle global, et la communication ne se fera plus. Une solution pour éviter ce type de comportements est d'imposer une équité forte entre les processus. Si infiniment souvent l'émetteur peut tirer une règle de transition, alors infiniment souvent l'émetteur tirera une règle de transition. On exprime cette propriété grâce à deux propositions atomiques E_{tirable} et E_{tiree} qui expriment respectivement qu'une transition (au moins) de E est tirable, et qu'une transition de E vient d'être tirée. Cette dernière proposition est codée dans les états de contrôle par une variable qui mémorise la dernière règle tirée. D'autre part, le fait qu'une règle de l'émetteur soit tirable, est exprimable en regardant le contenu des canaux, et plus précisément, le premier message de chaque canal. La formule d'équité pour l'émetteur s'écrit :

$$\Box\Diamond E_{\mathsf{tirable}} \rightarrow \Box\Diamond E_{\mathsf{tiree}}.$$

On écrit une formule similaire pour le récepteur :

$$\Box\Diamond R_{\mathsf{tirable}} \rightarrow \Box\Diamond R_{\mathsf{tiree}}.$$

Finalement l'équité entre les deux composants s'exprime par la conjonction des deux formules :

$$\Phi_c \stackrel{\text{def}}{=} \Box\Diamond E_{\mathsf{tirable}} \rightarrow \Box\Diamond E_{\mathsf{tiree}} \wedge \Box\Diamond R_{\mathsf{tirable}} \rightarrow \Box\Diamond R_{\mathsf{tiree}}.$$

Les deux hypothèses d'équité présentées ne sont pas encore suffisantes. Par exemple il se pourrait que dans l'état E_1R_1, le récepteur écrive toujours le message 1 sur le canal c_2 et que l'émetteur écrive de son côté le message 0 sur le canal c_1. Ce comportement respecte l'équité entre les composants exprimée par ϕ_c, mais on aimerait, pour avoir une progression dans le protocole, que le récepteur lise le message 0 dans c_1 plutôt qu'il écrive dans c_2. Nous rajoutons donc finalement une hypothèse d'équité pour privilégier les lectures par rapport aux écritures. On dénote par L_{tirable} l'ensemble des configurations où une lecture est tirable, et par L_{tiree} l'ensemble des configurations où une lecture vient d'être tirée. La priorité des lectures sur les écritures s'écrit alors de la façon suivante :

$$\phi_l \stackrel{\text{def}}{=} \Box\Diamond L_{\mathsf{tirable}} \rightarrow \Box\Diamond L_{\mathsf{tiree}}.$$

Finalement, notre hypothèse d'équité est la combinaison de ϕ_c et ϕ_l : $\phi_{eq} \stackrel{\text{def}}{=} \phi_c \wedge \phi_l$.

La propriété de progrès que l'on cherche à vérifier pour le Bit Alterné, exprime que les quatre états de contrôle sont visités infiniment souvent :

$$\varphi \stackrel{\text{def}}{=} \Box\Diamond E_1R_1 \wedge \Box\Diamond E_1R_2 \wedge \Box\Diamond E_2R_2 \wedge \Box\Diamond E_2R_1.$$

En modélisant le Bit Alterné par un NPLCS, le problème que nous considérons est de savoir si, quelque soit l'adversaire, la formule $\phi_e \rightarrow \varphi$ est vraie avec probabilité 1.

$$\forall \mathcal{U},\ \Pr_{\mathcal{U}}(\sigma_0 \models \phi_e \rightarrow \varphi) = 1.$$

Comme il est plus aisé de raisonner pour les adversaires sous la forme existentielle, nous transformons cette question :

$$\begin{aligned}\forall \mathcal{U},\ \Pr_{\mathcal{U}}(\sigma_0 \models \phi_e \rightarrow \varphi) = 1 \ \text{ ssi } \ &\neg\Big(\exists \mathcal{U},\ \Pr_{\mathcal{U}}(\sigma_0 \models \phi_e \wedge \neg\varphi) > 0\Big)\\ \text{ssi } \ &\neg\Big(\exists \mathcal{U},\ \Pr_{\mathcal{U}}(\sigma_0 \models \phi_e \wedge (\Diamond\Box\neg E_1R_1 \vee \Diamond\Box\neg E_1R_2 \vee \Diamond\Box\neg E_2R_2 \vee \Diamond\Box\neg E_2R_1)) > 0\Big).\end{aligned}$$

On peut à présent distribuer la conjonction pour obtenir une disjonction de conjonctions :

$$\begin{aligned}&\phi_e \wedge (\Diamond\Box\neg E_1R_1 \vee \Diamond\Box\neg E_1R_2 \vee \Diamond\Box\neg E_2R_2 \vee \Diamond\Box\neg E_2R_1)\\ &\quad\equiv (\phi_e \wedge \Diamond\Box\neg E_1R_1) \vee (\phi_e \wedge \Diamond\Box\neg E_1R_2) \vee (\phi_e \wedge \Diamond\Box\neg E_2R_2) \vee (\phi_e \wedge \Diamond\Box\neg E_2R_1)\end{aligned} \tag{6.1}$$

que l'on note $\phi_{1,1} \vee \phi_{1,2} \vee \phi_{2,2} \vee \phi_{2,1}$.

Maintenant, on utilise le fait qu'un adversaire $\mathcal{U}$ vérifie $\Pr_{\mathcal{U}}(\sigma_0 \models \phi_{1,1} \vee \phi_{1,2} \vee \phi_{2,2} \vee \phi_{2,1}) > 0$ si et seulement si il existe un couple (i, j) tel que $\Pr_{\mathcal{U}}(\sigma_0 \models \phi_{i,j}) > 0$. On peut argumenter que notre modélisation du protocole du Bit Alterné possède des propriétés de symétrie (les messages de l'émetteur correspondent aux accusés du récepteur par exemple) pour conclure que le choix du couple d'indices (i, j) n'a pas d'influence sur la réponse. La configuration initiale aurait bien une influence, vu que l'on fixe les états de contrôle de départ, mais nous allons montrer que la propriété souhaitée est vérifiée quelque soit la configuration initiale : c'est-à-dire que l'ensemble de configurations qui conviennent est Conf. Pour ces raisons on considère finalement le problème suivant :

$$\neg\Big(\exists \mathcal{U}\ \Pr_{\mathcal{U}}(\sigma \models \phi_{1,1}) > 0\Big)$$

soit encore

$$\neg\Big(\exists \mathcal{U}\ \Pr_{\mathcal{U}}\big(\sigma \models (\phi_e \wedge \Diamond\Box\neg E_1R_1)\big) > 0\Big).$$

Cette question peut être résolue en utilisant les techniques exposées dans le Chapitre 5. Grâce à notre outil, nous calculons l'ensemble T des configurations initiales σ pour lesquelles il n'existe aucun adversaire $\mathcal{U}$ tel que $\Pr_{\mathcal{U}}(\sigma_0 \models \phi_e \wedge \Diamond\Box\neg E_1R_1) > 0$.

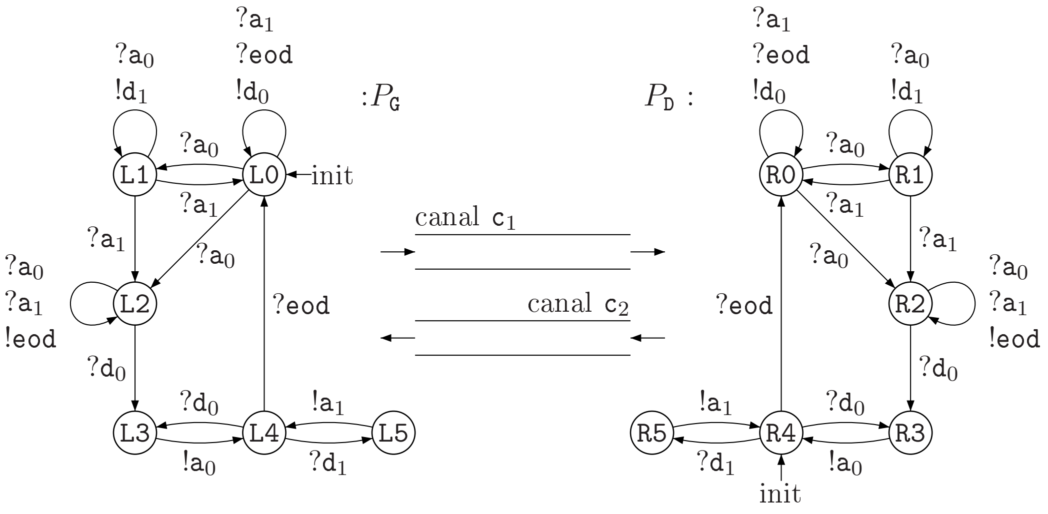

FIG. 6.3 – Le protocole de communication de Pachl

Il faut pour cela, scinder la formule selon les disjonctions de cas pour les propriétés d'équité, et en appliquer l'outil sur ces sous-formules. On obtient alors le résultat

$$T = \langle E_1R_1, \uparrow\varepsilon, \uparrow\varepsilon\rangle + \langle E_1R_2, \uparrow\varepsilon, \uparrow\varepsilon\rangle + \langle E_2R_2, \uparrow\varepsilon, \uparrow\varepsilon\rangle + \langle E_2R_1, \uparrow\varepsilon, \uparrow\varepsilon\rangle = \mathsf{Conf}.$$

Ceci montre que pour toute configuration initiale σ_0 et pour tout adversaire $\mathcal{U}$ on a $\Pr_{\mathcal{U}}(\sigma_0 \models \phi_e \rightarrow \varphi) = 1$.

Conclusion Sous les hypothèses d'équité que nous avons faites, à savoir sur les pertes, les composants, et les lectures, les comportement du protocole du Bit Alterné est correct, quel que soit l'environnement. Nous avons prouvé la vivacité du protocole du bit alterné sous réserves d'hypothèse d'équité sur les pertes (ici modélisées de façon probabiliste), d'équité entre les composants, et telles que les lectures soient favorisées.

6.3 Vérification automatique du protocole de Pachl

6.3.1 Présentation du protocole

Le protocole de Pachl [Pac87], comme le protocole du bit alterné, manipule des communications par des canaux non fiables et nous a servi de cas d'étude pour notre prototype. Il est constitué de deux entités identiques $P_{\mathtt{G(auche)}}$ et $P_{\mathtt{D(roit)}}$ qui échangent des données par des canaux pouvant perdre des messages. Le protocole utilise un mécanisme pour accuser la réception des messages qui est basé sur le protocole du bit alterné. La Figure 6.3 représente le protocole de Pachl. Les contenus précis des canaux sont abstraits dans ce protocole : on utilise simplement $\mathtt{d}_0, \mathtt{d}_1 \in \mathsf{M}$ pour représenter le bit de contrôle du message. Les messages $\mathtt{a}_0, \mathtt{a}_1 \in \mathsf{M}$ sont les accusés correspondants. Le protocole débute dans la configuration $(\mathtt{L0}, \mathtt{R4})$ où $P_{\mathtt{G}}$ est l'émetteur, et $P_{\mathtt{D}}$ le récepteur. À tout moment (sous réserve que le dernier message envoyé ait été accusé), l'émetteur peut signaler la fin de ses envois grâce au message $\mathtt{eod} \in \mathsf{M}$. Les deux entités échangent alors leur rôle (l'émetteur devient récepteur et vice versa). Le message eod ne porte pas de bit de contrôle, contrairement aux autres messages, et il n'est pas nécessaire que sa réception soit accusée par le récepteur.

Nous expliquons à présent comment ce protocole est modélisé par un LCS. La Figure 6.3 donne lieu à un automate communicant en formant le produit asynchrone des deux automates $P_{\mathtt{G}}$ et $P_{\mathtt{D}}$. En munissant cet automate communicant d'un taux de perte λ, on obtient un LCS $\mathcal{L}_{\text{Pachl}}$. $\mathcal{L}_{\text{Pachl}}$ possède $6 \times 6 = 36$ états de contrôle et $(18+18) \times 6 = 216$ règles de transitions. Pour pouvoir exprimer des événements comme «la règle de transition δ a été tirée», qui sont intrinsèques aux conditions d'équité, notre prototype nécessite l'ajout d'une variable d'historique qui prend pour valeur la dernière transition tirée. En pratique, seulement son étiquette est nécessaire, puisque les actions sont distinctes entre les deux entités. Cette variable multiplierait à nouveau le nombre d'états et de transitions par 20, mais tous les couples (état de contrôle,dernière action) n'ont pas forcément un sens, ce qui permet de réduire le modèle final à 144 états de contrôle et 948 règles de transition. Dans tous les résultats que nous donnerons ici, on n'utilise pas la dénomination de ces 144 états de contrôle, mais plutôt leur projection sur les 36 états du produit de $P_{\mathtt{G}}$ par $P_{\mathtt{D}}$.

6.3.2 Vérification automatique

Propriétés de sûreté Dans [Pac87] Pachl a calculé à la main l'ensemble $\mathit{Post}^*(\mathtt{Init})$ de toutes les configurations accessibles dans $\mathcal{L}_{\text{Pachl}}$ à partir de la configuration initiale vide $\mathtt{Init} = (\mathtt{L0}, \mathtt{R4}, \varepsilon, \varepsilon)$. Ce genre de calcul d'accessibilité en avant peut parfois être fait automatiquement grâce aux techniques décrites dans [ACBJ04], même si la terminaison des calculs en avant n'est pas garantie en général. Dans le cas particulier du protocole de Pachl, ces calculs montrent que le protocole préserve l'intégrité de la communication dans le sens où les pertes ne peuvent pas introduire de confusion entre les messages de données.

En comparaison avec les calculs en avant, nos algorithmes pour les régions vont dans le sens d'un calcul en arrière, pour lequel la terminaison est garantie. Notre implémentation permet de calculer automatiquement l'ensemble des configurations pour lesquelles aucune règle de transition n'est tirable (configurations de *deadlock*).

$$\mathtt{Dead} \stackrel{\text{def}}{=} \mathsf{Conf} \setminus \mathit{Pre}(\mathsf{Conf}) = \langle \mathtt{L4}, \mathtt{R4}, \varepsilon, \varepsilon \rangle.$$

Heureusement, $\mathtt{Dead}$ n'est pas atteignable à partir de $\mathtt{Init}$, qui est la configuration initiale du protocole. On peut calculer l'ensemble des prédécesseurs de $\mathtt{Dead}$: $\mathit{Pre}^*(\mathtt{Dead})$ qui est l'ensemble des configurations non sûres, au sens où elles peuvent mener à une configuration bloquée. En intersectant avec $\uparrow\mathtt{Init}$ (la région de contrôle ayant $\mathtt{Init}$ pour état de contrôle) on obtient l'ensemble des contenus de canaux initiaux qui sont non sûrs :

$$\begin{aligned} \mathit{Pre}^*(\mathtt{Dead}) \sqcap \uparrow\mathtt{Init} = \\ \langle \mathtt{L0}, \mathtt{R4}, \uparrow\varepsilon, \uparrow\mathtt{a}_0\mathtt{d}_0 \rangle + \langle \mathtt{L0}, \mathtt{R4}, \uparrow\mathtt{eod}\,\mathtt{a}_0, \uparrow\mathtt{a}_0 \rangle + \langle \mathtt{L0}, \mathtt{R4}, \uparrow\mathtt{d}_0\mathtt{eod}\,\mathtt{a}_0, \uparrow\varepsilon \rangle. \end{aligned}$$

Ainsi, un blocage est possible à partir de l'état de contrôle $(\mathtt{L0}, \mathtt{R4})$ si les canaux ont initialement des contenus inappropriés. Par exemple, soit $(L_0, R_4, \mathtt{d}_0\mathtt{eod}\,\mathtt{a}_0, \varepsilon)$ une configuration initiale. Le chemin suivant mène à une configuration bloquée :

$$\begin{aligned} &(L_0, R_4, \mathtt{d}_0\mathtt{eod}\,\mathtt{a}_0, \varepsilon) \xrightarrow{c_1?\mathtt{d}_0}_{\text{perf}} (L_0, R_3, \mathtt{eod}\,\mathtt{a}_0, \varepsilon) \xrightarrow{c_2!\mathtt{a}_0}_{\text{perf}} (L_0, R_3, \mathtt{eod}\,\mathtt{a}_0, \mathtt{a}_0) \\ &\xrightarrow{c_1!\mathtt{d}_0}_{\text{perf}} (L_0, R_4, \mathtt{eod}\,\mathtt{a}_0\mathtt{d}_0, \mathtt{a}_0) \xrightarrow{c_2?\mathtt{a}_0}_{\text{perf}} (L_2, R_4, \mathtt{eod}\,\mathtt{a}_0\mathtt{d}_0, \varepsilon) \xrightarrow{c_1?\mathtt{eod}}_{\text{perf}} (L_2, R_0, \mathtt{a}_0\mathtt{d}_0, \varepsilon) \\ &\xrightarrow{c_2!d_0}_{\text{perf}} (L_2, R_0, \mathtt{a}_0\mathtt{d}_0, \mathtt{d}_0) \xrightarrow{c_1?\mathtt{a}_0}_{\text{perf}} (L_2, R_2, \mathtt{d}_0, \mathtt{d}_0) \xrightarrow{c_1?\mathtt{d}_0}_{\text{perf}} (L_2, R_3, \varepsilon, \mathtt{d}_0) \\ &\xrightarrow{c_2!\mathtt{a}_0} (L_2, R_4, \varepsilon, \mathtt{d}_0) \xrightarrow{c_2?\mathtt{d}_0}_{\text{perf}} (L3, R_4, \varepsilon, \varepsilon) \xrightarrow{c_1!\mathtt{a}_0} (L_4, R_4, \varepsilon, \varepsilon) \end{aligned}$$

Propriétés de vivacité On peut maintenant en venir à ce qui était la motivation principale de notre étude : prouver des propriétés de progrès même si cela exige des hypothèses d'équité.

Dans cette étude de cas, le problème que nous traitons est en général de calculer l'ensemble des toutes les configurations satisfaisant $\mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond A) = 1$ pour tous les adversaires $\mathcal{U}$ équitables pour une condition d'équité $\mathcal{F}$.

Grâce aux algorithmes de la section 5.4, ceci est lié au calcul de $\mathbf{E}[\Box\overline{A}]^{\mathcal{F}}$. Plus précisément :

$$\begin{aligned}\{\sigma \mid \forall\mathcal{U}\mathcal{F}\text{-fair } \mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond A) = 1\} &= \mathsf{Conf} \setminus \{\sigma \mid \exists\mathcal{U}\mathcal{F}\text{-fair } \mathrm{Pr}_{\mathcal{U}}(\sigma \models \Box\Diamond A) < 1\} \\ &= \mathsf{Conf} \setminus \{\sigma \mid \exists\mathcal{U}\mathcal{F}\text{-fair } \mathrm{Pr}_{\mathcal{U}}(\sigma \models \Diamond\Box\overline{A}) > 0\} \\ &= \mathsf{Conf} \setminus \mathit{Pre}^*(\mathbf{E}[\Box\overline{A}]^{\mathcal{F}}).\end{aligned}$$

La dernière égalité provient du Théorème 5.74 (d) (cf. page 121).

Le calcul de $\mathbf{E}[\Box\overline{A}]^{\mathcal{F}}$ nécessite de considérer tous les sous-ensembles de $\mathcal{F}$ ce qui entraîne une explosion combinatoire pour des ensemble $\mathcal{F}$ de cardinal élevé. Nous n'avons pas pour le moment développé et implémenté des heuristiques pour surmonter cette difficulté. C'est pourquoi les exemples que nous traitons dans cette étude préliminaire, et que nous présentons ici, ont toujours des «petits» $\mathcal{F}$, c'est-à-dire ayant une nombre restreint d'ensembles d'équité, chacun d'entre eux pouvant avoir un grand nombre de règles de transitions. Par exemple, nous nous sommes intéressés à l'équité forte entre les processus $\mathcal{F}_{\texttt{process}} = \{F_{\texttt{left}}, F_{\texttt{right}}\}$, où $F_{\texttt{left}}$ est composé de l'ensemble des règles dont le processus gauche est responsable (et symétriquement pour $F_{\texttt{left}}$). Nous avons également considéré une forme d'équité pour les lectures : $\mathcal{F}_{\texttt{read}} = \{F_{\texttt{read}}\}$ pour laquelle $F_{\texttt{read}}$ contient exactement les règles de transitions qui sont des lectures.

Quant à la propriété à vérifier, nous avons considéré par exemple les questions du type : «une transition δ est-elle tirée infiniment souvent ?» (en utilisant la variable d'historique) ou «un processus change-t-il infiniment souvent d'état de contrôle ?», etc...

Puisque qu'une conjonction $\mathrm{Pr}_{\mathcal{U}}(s \models \Box\Diamond A_i) = 1$ (avec quantification universelle sur les adversaires $\mathcal{U}$) donne $\mathrm{Pr}_{\mathcal{U}}(s \models \bigwedge_i \Box\Diamond A_i) = 1$, on peut vérifier des formule du type $\bigwedge_i \Box\Diamond\texttt{Li} \wedge \bigwedge_i \Box\Diamond\texttt{Ri}$, qui exprime le progrès dans la communication entre les deux processus.

Nous avons étudié trois cas particuliers pour les hypothèses d'équité sur les adversaires. Tout d'abord, on considère l'hypothèse d'équité $\mathcal{F} = \mathcal{F}_{\texttt{read}}$ constituée d'un unique ensemble, celui des règles de lecture, et l'ensemble $A = \mathit{After}_{\texttt{left}}$ des configurations pour lesquelles une action du processus gauche vient d'être tiré. Cet ensemble peut être défini puisque nous mémorisons dans les états de contrôle la dernière transition tirée.

Le calcul $\mathsf{Conf} \setminus \mathit{Pre}^*(\mathbf{E}[\Box\overline{A}]^{\mathcal{F}})$ donne un ensemble de configurations qui contient la configuration initiale `Init` (sans pour autant contenir toutes les configurations qui ont $L0R4$ comme état global). Cela signifie que tous les adversaires équitables pour les lectures (*i.e.* pour lesquels presque toutes les exécutions des lectures sont tirées infiniment souvent, si elles sont infiniment tirables) le composant gauche effectuera une infinité d'actions avec probabilité 1.

Cependant, cette conclusion ne permet pas de certifier que le composant gauche change d'état de contrôle infiniment souvent presque sûrement. En effet, le NPLCS qui modélise le protocole de Pachl possède des boucles (de lecture ou d'écriture) sur plusieurs états. On raffine donc l'étude en considérant à présent pour A l'ensemble des configurations qui suivent un véritable changement d'état de $P_{\texttt{G}}$. Pour exprimer cet ensemble de configurations, on utilise encore le fait que la dernière transition tirée est mémorisée. Il suffit alors de spécifier les transitions qui font changer $P_{\texttt{G}}$ d'état de contrôle. Écrivons $A = \mathit{After}_{\texttt{left-move}}$. La réponse de notre outil est encore une fois l'appartenance de la configuration initiale $\texttt{Init}$ à $\mathsf{Conf} \setminus Pre^*(\mathbf{E}[\Box\overline{A}]^{\mathcal{F}})$, ce qui signifie que si les lectures ne sont pas négligées, pour tous les adversaires, le composant gauche change d'état infiniment souvent presque sûrement.

Enfin, nous avons fait une dernière étude avec un ensemble $\mathcal{F}$ qui n'est pas un singleton. Pour $\mathcal{F} = \{F_{\texttt{read}}, F_{\texttt{right-read}}\}$, composé d'une part des transitions de lectures, et d'autre part des lectures du composant $P_{\texttt{D}}$, et pour l'ensemble $A = \mathit{After}_{\texttt{left}}$ de configurations, on obtient à nouveau

$$\texttt{Init} \in \mathsf{Conf} \setminus Pre^*(\mathbf{E}[\Box\overline{A}]^{\mathcal{F}}).$$

À partir de $\texttt{Init}$, l'ensemble des configurations A sera visité infiniment souvent presque sûrement, quel que soit l'adversaire équitable pour $\mathcal{F}$.

Conclusion Pour le protocole de Pachl, nous avons dans un premier temps calculé l'ensemble des configurations bloquées, et de leur prédécesseurs, c'est-à-dire des configurations qui peuvent mener à une configuration bloquée. Ceci permet de connaître les configurations que l'on peut choisir comme configuration initiale, en étant sûr qu'aucun blocage ne surviendra. Cet ensemble de configurations était très fastidieux à calculer à la main, même si le protocole de Pachl est relativement petit.

Nous avons également mené une étude de vivacité pour le protocole de Pachl. Pour cela, nous avons considéré des adversaires équitables, et comme propriété de vivacité, le passage infiniment souvent dans chacun des états de contrôle des deux composants. Cette propriété permet d'exprimer que la communication se fait entre les deux composants, et que le système ne reste pas à jamais dans le même état de contrôle global. Pour des petits ensembles d'hypothèses d'équité, nous avons pu prouver ce bon fonctionnement.

Conclusion

Ce travail de recherche avait pour but d'étudier les systèmes à canaux à pertes sous un angle probabiliste. En conclusion, nous récapitulons les résultats obtenus et donnons quelques perspectives.

Nous avons introduit et étudié deux types de modèles pour les LCS probabilistes : les LCS uniquement probabilistes (PLCS - *Probabilistic Channel Systems*) et les LCS à la fois non déterministes et probabilistes (NPLCS - *Nondeterministic and Probabilistic Channel Systems*). Pour ces deux modèles nous avons présenté des techniques pour la vérification qualitative de propriétés (allant des propriétés de sûreté ou de vivacité aux propriétés générales exprimées par des formules du temps linéaire). La méthode générale pour résoudre ces questions de vérification est de les réduire à des questions d'accessibilité plus ou moins compliquées et plus ou moins nombreuses sur le LCS sous-jacent. On donne dans chaque cas des caractérisations indépendantes du taux de pertes, et qui repose uniquement sur la structure de l'automate communicant. Pour résoudre les diverses questions d'accessibilité dans les LCS, nous avons établit un théorème de convergence des points fixes dans le cadre général des systèmes infinis munis d'un bel ordre. C'est ce résultat que nous appliquons dans le cas des LCS, pour la vérification qualitative des PLCS et de NPLCS.

Regular model-checking des systèmes infinis Pour la vérification de systèmes infinis, et plus particulièrement de systèmes munis d'un bel ordre, nous avons présenté un critère de convergence en temps fini de points fixe. Pour cela, on s'appuie sur des notions usuelles d'algèbre de régions, et en utilisant les bonnes propriétés du bel ordre on exhibe un critère de convergence pour les termes d'un μ-calcul sur l'algèbre des régions. Une condition suffisante est que les termes soient gardés, c'est-à-dire que les variables liées soient sous la portée d'opérateurs de fermeture vers le haut (resp. vers le bas) lorsque l'on considère un plus petit point fixe (resp, plus grand point fixe). Ce résultat général peut être appliqué à de nombreux exemples de systèmes infinis munis d'un bel ordre, et s'applique en particulier aux *Lossy Channel Systems*, pour lesquelles le bel ordre est induit par la relation de sous-mot pour les contenus des canaux.

Nous avons par ailleurs défini une extension des *Lossy Channel Systems*, appelée «LCS étendus». Dans ce nouveau modèle, on ajoute la possibilité de munir les règles de transitions de gardes régulières. Ces gardes permettent d'autoriser ou non une transition selon les contenus des canaux de la configuration de départ. Le théorème de convergence des termes gardés du μ-calcul peut être appliqué à des expressions définies pour les LCS étendus.

À la fois pour les LCS et les LCS étendus, notre théorème de convergence permet de prouver de façon élégante la convergence de points fixes, et donc retrouver ou découvrir la calculabilité de certains ensembles (comme l'ensemble des prédécesseurs, des prédécesseurs contraints, ou des prédécesseurs sans risque par rapport à une région).

LCS purement probabilistes Pour les LCS purement probabilistes, nous avons proposé un nouveau modèle améliorant les précédents. Notre modèle est encore sous forme de chaîne de Markov, mais les probabilités de chaque transition sont quelques peu différentes. Il se distingue de l'ancien modèle appelé modèles à pertes globales par les probabilité attribuées aux pertes de messages. Notre modèle est plus réaliste puisque, comme ce que l'on observe en pratique, il a pour conséquence que les pertes sont plus probables lorsque de nombreux messages sont en transit. D'autre part, les questions de vérification sur ce modèle sont indépendantes du taux de pertes. Ceci est souhaitable car bien souvent, ce taux ne peut être correctement évalué et est fixé de façon arbitraire.

Les techniques de vérification des LCS purement probabilistes reposent sur un fait crucial : l'existence d'un attracteur fini dans la chaîne de Markov associée. Cela signifie que parmi les configurations, on peut exhiber un ensemble fini qui est visité infiniment souvent presque sûrement. Le fait que presque toutes les exécutions visiteront infiniment souvent cet ensemble fini, permet de simplifier les questions d'accessibilité et d'accessibilité répétée, et de les rapprocher du cas, bien connu à présent, où la chaîne de Markov est finie.

Pour prouver l'existence de l'attracteur fini dans les chaînes de Markov induites par les LCS probabilistes, nous appliquons un critère général aux chaînes de Markov à espace d'états dénombrable. Ainsi, pour des familles entières de chaînes de Markov (orientées à gauche ou presque orientée à gauche), on prouve l'existence d'un attracteur fini. Ce critère est applicable pour les LCS probabilistes, et permet de montrer que l'ensemble des configurations avec canaux vides est un attracteur.

On traite alors la vérification de propriétés d'accessibilité ou d'accessibilité répétée pour les LCS purement probabilistes, comme un cas particulier de ces mêmes questions pour les chaînes de Markov avec attracteur. Les conclusions de cette étude sont la décidabilité de la vérification qualitative de propriétés d'accessibilité et d'accessibilité répétée pour les LCS purement probabilistes. Une approche classique (par produit de la chaîne de Markov avec un automate de Büchi déterministe) permet alors de généraliser ces résultats de décidabilité aux formules du temps linéaire.

LCS probabilistes et non déterministes Le modèle markovien des Lossy Channel Systems n'était pas pleinement satisfaisant. En effet, pour de nombreuses raisons, le non-déterminisme est essentiel dans certains systèmes, et ne peut pas être remplacé par un comportement aléatoire. C'est pourquoi nous avons développé un nouveau modèle pour les LCS : celui des LCS probabilistes et non déterministes. Dans ce modèle, seules les pertes de messages sont probabilistes, et le choix entre les actions est non déterministe. Le formalisme mathématique sous-jacent est alors celui des processus de décision markoviens, appelés aussi chaînes de Markov concurrentes.

Le modèle des processus de décision markoviens mélange probabilités et non déterminisme, ce qui permet de modéliser de façon plus fidèle les systèmes, mais d'un autre côté complexifie les problèmes de vérification. Un adversaire résout les choix non déterministes : il peut en particulier modéliser un environnement que l'on ne maîtrise pas. Lorsque l'adversaire est fixé, on obtient une chaîne de Markov, qui comme c'était le cas pour les PLCS, possède un attracteur fini. Les problèmes de vérification s'énoncent ainsi : «pour tous les adversaires possibles, a-t-on ϕ vérifiée presque sûrement ?».

Pour des formules du temps linéaire, et des adversaires généraux, on a montré que ce problème était indécidable. Néanmoins, on peut recouvrer la décidabilité pour une sous-classe importante des adversaires, celle des adversaires à mémoire finie. Ici encore, pour prouver la décidabilité d'une question, nous montrons que l'ensemble des configurations satisfaisantes peut être exprimé par un terme du μ-calcul gardé. Si les propositions atomiques des formules sont des régions, le résultat général de convergence des points fixes s'applique, et entraîne la décidabilité. De cette façon on montre que pour des adversaires à mémoire finie, la vérification qualitative de formules du temps linéaire pour les NPLCS est décidable. Ce résultat s'étend lorsqu'on ajoute des contraintes d'équité sur les adversaires.

La complexité des problèmes évoqués ci-dessus est non primitive récursive. Nous avons néanmoins implémenté les algorithmes de vérification en un prototype d'outil en Caml. Malgré une complexité théorique importante, cet outil nous a permis de vérifier des propriétés de bon fonctionnement pour deux protocoles : le protocole du Bit Alterné, et le protocole de Pachl. Pour ces deux protocoles, nous avons prouvé, en faisant des hypothèses d'équité (intégrées ou non à l'adversaire), une forme de progression : les deux composants visitent infiniment souvent chacun de leur états de contrôle.

Perspectives Le résultat général de convergence des points fixes possède des applications plus vastes que l'étude des PLCS et NPLCS. Nous avons par exemple commencé à l'appliquer au calcul des configurations gagnantes pour des jeux sur des Lossy Channel Systems. Au delà des applications à des variantes de LCS, il serait intéressant de regarder les questions traitables pour d'autres systèmes infinis munis d'un bel ordre, comme les réseaux de Petri par exemple.

Pour ce qui est du modèle purement probabiliste des LCS, une progression naturelle est de considérer des questions qualitatives (et non plus quantitatives). Quelques travaux ont été faits dans ce sens, mais de nombreuses questions sont encore ouvertes.

De la même façon, la vérification quantitative pour les NPLCS est une suite naturelle de notre travail. Sur ce sujet, à notre connaissance, aucun travail n'a été réalisé. Cependant, vu les difficultés rencontrées pour l'étude quantitative des PLCS, le problème de la vérification quantitative des NPLCS est certainement ardu. Une autre piste de recherche est la combinaison de plusieurs types de données, en plus des canaux de communication. On peut par exemple considérer un modèle pour combinant des files (à pertes ou non) et des compteurs, voire des horloges. Ces extensions permettraient une modélisation encore plus fidèle de certains protocoles de communication asynchrones.

Enfin, le prototype que nous avons développé gagnerait à être amélioré. Un véritable outil de model-checking permettrait d'une part de traiter des exemples plus conséquents, et d'autre part d'utiliser pour les ensembles de configurations de régions régulières plutôt que les ensembles symboliques.

Table des figures

Index

Table des notations

Nous utilisons les notations suivantes tout au long de cette thèse :

$\mathcal{M} = (E, \mathbf{P})$	chaîne de Markov
E	ses états de contrôle
$\mathbf{P}$	matrice de probabilités
$\mathsf{ST}_{\mathcal{M}}$	système de transitions associé à $\mathcal{M}$
$\mathbb{E}$	espérance
$\mathbb{P}$	probabilité pour un ens. de chemins de vérifier phi
niv	niveau
$\mathtt{PDM} = (E, \Delta)$	Processus de décision markovien
LCS	
$\mathcal{L} = (S, \mathsf{C}, \mathsf{M}, \Delta)$	LCS
$S = \{s, s', t, \cdots\}$	ensemble fini d'états
$A, B, \cdots \subseteq S$	région de contrôle
$\mathsf{C} = \{c, d, c_1, \cdots\}$	ensemble fini de canaux
$\mathsf{M} = \{m, m_1 \cdots\}$	ensemble de messages
$\mathsf{M}^* = \{v, x, y \cdots\}$	contenu d'un canal
$\mathsf{M}^{*\mathsf{C}} = \{\mathbf{w}, \mathbf{v}\}$	contenus de canaux
$\mathsf{Conf} = S \times \mathsf{M}^{*\mathsf{C}}$	ensemble des configurations
$\Delta = \{\delta\}$	fonction de transition
$\sigma = (s, \mathbf{w}),\ \tau = (t, \mathbf{v})$	configurations
$\Delta(\sigma)$	transitions tirables dans σ
l	fonction d'étiquetage : $S \to \Sigma$ et $\mathsf{Conf} \to \Sigma$
$\mathsf{ST}_{\mathcal{L}}$	système de transition associé à $\mathcal{L}$
PLCS	
$\mathcal{P} = (S, \mathsf{C}, \mathsf{M}, \Delta, \lambda, w)$	PLCS
w	fonction de poids
λ	taux de perte (et aussi λ_D, λ_I, λ_C)
$\mathcal{M}_{\mathcal{P}}$	chaîne de Markov associée à $\mathcal{P}$
NPLCS	
$\mathcal{N} = (S, \mathsf{C}, \mathsf{M}, \Delta, \lambda)$	NPLCS
$\mathcal{U}$, $\mathcal{V}$, $\mathcal{W}$	adversaires

Bibliographie

[ABd03] P. A. Abdulla, A. Bouajjani, and J. d'Orso. Deciding monotonic games. In *Proc. 17th Int. Workshop Computer Science Logic (CSL '03) and 8th Kurt Gödel Coll. (KGL '03), Vienna, Austria, Aug. 2003*, volume 2803 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2003.

[ABM05] P. Abdulla, N. Ben Henda, and R. Mayr. Verifying infinite Markov chains with a finite attractor or the global coarseness property. In *Proc. 20th IEEE Symp. Logic in Computer Science (LICS '05), Chicago, IL, USA, June 2005*, pages 127–136. IEEE Comp. Soc. Press, 2005.

[ABMS06] P. A. Abdulla, N. Ben Henda, R. Mayr, and S. Sandberg. Eager Markov chains. Technical Report 2006-009, Department of Information Technology, Uppsala University, March 2006.

[ABPJ00] P. A. Abdulla, C. Baier, S. Purushothaman Iyer, and B. Jonsson. Reasoning about probabilistic lossy channel systems. In *Proc. 11th Int. Conf. Concurrency Theory (CONCUR '00), University Park, PA, USA, Aug. 2000*, volume 1877 of *Lecture Notes in Computer Science*, pages 320–333. Springer, 2000.

[ABPJ05] P. A. Abdulla, C. Baier, S. Purushothaman Iyer, and B. Jonsson. Simulating perfect channels with probabilistic lossy channels. *Information and Computation*, 197(1–2) :22–40, 2005.

[ABRS05] P. A. Abdulla, N. Bertrand, A. Rabinovich, and Ph. Schnoebelen. Verification of probabilistic systems with faulty communication. *Information and Computation*, 202(2) :141–165, 2005.

[ABS01] A. Annichini, A. Bouajjani, and M. Sighireanu. TReX : A tool for reachability analysis of complex systems. In *Proc. 13th Int. Conf. Computer Aided Verification (CAV '01), Paris, France, July 2001*, volume 2102 of *Lecture Notes in Computer Science*, pages 368–372. Springer, 2001.

[ACBJ04] P. A. Abdulla, A. Collomb-Annichini, A. Bouajjani, and B. Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1) :39–65, 2004.

[AJ93] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. In *Proc. 8th IEEE Symp. Logic in Computer Science (LICS '93), Montreal, Canada, June 1993*, pages 160–170. IEEE Comp. Soc. Press, 1993.

[AJ96a] P. A. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. *Information and Computation*, 130(1) :71–90, 1996.

[AJ96b] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2) :91–101, 1996.

[AN01] A. Arnold and D. Niwiński. *Rudiments of μ-calculus*, volume 146 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science, 2001.

[AR03] P. A. Abdulla and A. Rabinovich. Verification of probabilistic systems with faulty communication. In *Proc. 6th Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS '03), Warsaw, Poland, Apr. 2003*, volume 2620 of *Lecture Notes in Computer Science*, pages 39–53. Springer, 2003.

[BA95] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. 15th Conf. Found. of Software Technology and Theor. Comp. Sci. (FST&TCS '95), Bangalore, India, Dec. 1995*, volume 1026 of *Lecture Notes in Computer Science*, pages 499–513. Springer, 1995.

[BBS06a] C. Baier, N. Bertrand, and Ph. Schnoebelen. A note on the attractor-property of infinite-state Markov chains. *Information Processing Letters*, 97(2) :58–63, 2006.

[BBS06b] C. Baier, N. Bertrand, and Ph. Schnoebelen. On computing fixpoints in well-structured regular model checking, with applications to lossy channel systems. In *Proc. 13th Int. Conf. on Logic for Programming and Artificial Intelligence, and Reasoning (LPAR '06), Phnom Penh, Cambodia, Nov. 2006*, volume 4246 of *Lecture Notes in Artificial Intelligence*, pages 347–361. Springer, 2006.

[BBS06c] C. Baier, N. Bertrand, and Ph. Schnoebelen. Symbolic verification of communicating systems with probabilistic message losses : liveness and fairness. In *Proc. 26th IFIP WG6.1 Int. Conf. Formal Techniques for Networked and Distributed Systems (FORTE '06), Paris, France, Sep. 2006*, volume 4229 of *Lecture Notes in Computer Science*, pages 212–227. Springer, 2006.

[BBS07] C. Baier, N. Bertrand, and Ph. Schnoebelen. Verifying nondeterministic probabilistic channel systems against ω-regular linear-time properties. *ACM Transactions on Computational Logic*, 2007. http ://arxiv.org/abs/cs.LO/0511023. To appear.

[BD87] S. Budkowski and P. Dembinski. An introduction to Estelle : A specification language for distributed systems. *Computer Networks and ISDN Systems*, 14(3), 1987.

[BE99] C. Baier and B. Engelen. Establishing qualitative properties for probabilistic lossy channel systems : an algorithmic approach. In *Proc. 5th Int. AMAST Workshop Formal Methods for Real-Time and Probabilistic Systems (ARTS '99), Bamberg, Germany, May 1999*, volume 1601 of *Lecture Notes in Computer Science*, pages 34–52. Springer, 1999.

[BFLS05] S. Bardin, A. Finkel, J. Leroux, and Ph. Schnoebelen. Flat acceleration in symbolic model checking. In *Proc. 3rd Int. Symp. Automated Technology for Verification and Analysis (ATVA '05), Taipei, Taiwan, Oct. 2005*, volume 3707 of *Lecture Notes in Computer Science*, pages 474–488. Springer, 2005.

[BG99] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. *Formal Methods in System Design*, 14(3) :237–255, 1999.

[BGWW97] B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of QDDs (extended abstract). In *Proc. 4th Int. Symp. Static Analysis (SAS '97), Paris, France, Sep. 1997*, volume 1302 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 1997.

[BH97] A. Bouajjani and P. Habermehl. Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations. In *Proc. 24th Int. Coll. Automata, Languages, and Programming (ICALP '97), Bologna, Italy, July 1997*, volume 1256 of *Lecture Notes in Computer Science*, pages 560–570. Springer, 1997.

[BHV04] A. Bouajjani, P. Habermehl, and T. Vojnar. Abstract regular model checking. In *Proc. 16th Int. Conf. Computer Aided Verification (CAV '04), Boston, MA, USA, July 2004*, volume 3114 of *Lecture Notes in Computer Science*, pages 372–386. Springer, 2004.

[BJNT00] A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *Proc. 12th Int. Conf. Computer Aided Verification (CAV '00), Chicago, IL, USA, July 2000*, volume 1855 of *Lecture Notes in Computer Science*, pages 403–418. Springer, 2000.

[BK98] C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3) :125–155, 1998.

[BLW03] B. Boigelot, A. Legay, and P. Wolper. Iterating transducers in the large (extended abstract). In *Proc. 15th Int. Conf. Computer Aided Verification (CAV '03), Boulder, CO, USA, July 2003*, volume 2725 of *Lecture Notes in Computer Science*, pages 223–235. Springer, 2003.

[Boi03] B. Boigelot. On iterating linear transformations over recognizable sets of integers. *Theoretical Computer Science*, 309(1–3) :413–468, 2003.

[Bré99] P. Brémaud. *Markov chains : Gibbs fields, Monte Carlo simulation, and queues*, volume 31 of *Texts in Applied Mathematics*. Springer, 1999.

[BS03] N. Bertrand and Ph. Schnoebelen. Model checking lossy channels systems is probably decidable. In *Proc. 6th Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS '03), Warsaw, Poland, Apr. 2003*, volume 2620 of *Lecture Notes in Computer Science*, pages 120–135. Springer, 2003.

[BS04] N. Bertrand and Ph. Schnoebelen. Verifying nondeterministic channel systems with probabilistic message losses. In Ramesh Bharadwaj, editor, *Proc. 3rd Int. Workshop on Automated Verification of Infinite-State Systems (AVIS '04), Barcelona, Spain, Apr. 2004*, 2004.

[BSW69] K. Bartlett, R. Scantlebury, and P. Wilkinson. A note on reliable full-duplex transmission over half-duplex links. *Communications of the ACM*, 12(5) :260–261, 1969.

[BZ83] D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2) :323–342, 1983.

[CFP96] G. Cécé, A. Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1) :20–31, 1996.

[CMP92] E. Chang, Z. Manna, and A. Pnueli. Characterization of temporal property classes. In *Proc. 19th Int. Coll. Automata, Languages, and Programming (ICALP '92), Vienna, Austria, July 1992*, volume 623 of *Lecture Notes in Computer Science*, pages 474–486. Springer, 1992.

[CY95] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4) :857–907, 1995.

[dA99] L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In *Proc. 10th Int. Conf. Concurrency Theory (CONCUR '99), Eindhoven, The Nederlands, Aug. 1999*, volume 1664 of *Lecture Notes in Computer Science*, pages 66–81, 1999.

[EC80] E. A. Emerson and E. M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proc. 7th Coll. Automata, Languages and Programming (ICALP '80), Noordwijkerhout, NL, Jul. 1980*, volume 85 of *Lecture Notes in Computer Science*, pages 169–181. Springer, 1980.

[Fin94] A. Finkel. Decidability of the termination problem for completely specificied protocols. *Distributed Computing*, 7(3) :129–135, 1994.

[FM96] A. Finkel and O. Marcé. Verification of infinite regular communicating automata. Internal Report 96-4, Laboratoire d'Informatique Fondamentale et Appliquée de Cachan, ENS Cachan, France, April 1996.

[GTW02] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, logics, and infinite games : a guide to current research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc. (3)*, 2(7) :326–336, 1952.

[HSP83] S. Hart, M. Sharir, and A. Pnueli. Termination of probabilistic concurrent programs. *ACM Trans. Programming Languages and Systems*, 5(3) :356–380, 1983.

[HV05] P. Habermehl and T. Vojnar. Regular model checking using inference of regular languages. In *Proc. 6th Int. Workshop on Verification of Infinite State Systems (INFINITY '04), London, UK, Sep. 2004*, volume 138(3) of *Electronic Notes in Theor. Comp. Sci.*, pages 21–36. Elsevier Science, 2005.

[ITU99] Specification and description language (SDL). ITU-T Recommendation Z.100, International Telecommunications Union, November 1999.

[KMM+01] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. *Theoretical Computer Science*, 256(1–2) :93–112, 2001.

[Kru72] J. B. Kruskal. The theory of well-quasi-ordering : A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, 13(3) :297–305, 1972.

[KS06] A. Kučera and Ph. Schnoebelen. A general approach to comparing infinite-state systems with their finite-state specifications. *Theoretical Computer Science*, 358(2–3) :315–333, 2006.

[May03] R. Mayr. Undecidable problems in unreliable computations. *Theoretical Computer Science*, 297(1–3) :337–354, 2003.

[Ouv98] J.Y. Ouvrard. *Probabilités (Tome 1)*. Cassini, 1998.

[Ouv00] J.Y. Ouvrard. *Probabilités (Tome 2)*. Cassini, 2000.

[OW06] J. Ouaknine and J. Worrell. On metric temporal logic and faulty Turing machines. In *Proc. 9th Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS '06), Vienna, Austria, Apr. 2006*, volume 3921 of *Lecture Notes in Computer Science*, pages 217–230. Springer, 2006.

[Pac87] J. K. Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *Proc. 7th IFIP WG6.1 Int. Workshop on Protocol Specification, Testing, and Verification (PSTV '87), Zurich, Switzerland, May 1987*, pages 207–219. North-Holland, 1987.

[Pan01] P. Panangaden. Measure and probability for concurrency theorists. *Theoretical Computer Science*, 253(2) :287–309, 2001.

[Per90] D. Perrin. Finite automata. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 1, pages 1–57. Elsevier Science, 1990.

[PN97] S. Purushothaman Iyer and M. Narasimha. Probabilistic lossy channel systems. In *Proc. 7th Int. Joint Conf. Theory and Practice of Software Development (TAPSOFT '97), Lille, France, Apr. 1997*, volume 1214 of *Lecture Notes in Computer Science*, pages 667–681. Springer, 1997.

[Put94] M. L. Puterman. *Markov decision processes : discrete stochastic dynamic programming.* John Wiley & Sons, 1994.

[PZ86] A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1 :53–72, 1986.

[Rab06] A. Rabinovich. Quantitative analysis of probabilistic lossy channel systems. *Information and Computation*, 204(5) :713–740, 2006.

[Sch01] Ph. Schnoebelen. Bisimulation and other undecidable equivalences for lossy channel systems. In *Proc. 4th Int. Symp. Theoretical Aspects of Computer Software (TACS '01), Sendai, Japan, Oct. 2001*, volume 2215 of *Lecture Notes in Computer Science*, pages 385–399. Springer, 2001.

[Sch02] Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5) :251–261, 2002.

[Sch04] Ph. Schnoebelen. The verification of probabilistic lossy channel systems. In C. Baier et al., editors, *Validation of Stochastic Systems – A Guide to Current Research*, volume 2925 of *Lecture Notes in Computer Science*, pages 445–465. Springer, 2004.

[Tho90] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 4, pages 133–191. Elsevier Science, 1990.

[Var85] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th Symp. Foundations of Computer Science (FOCS '85), Portland, OR, USA, Oct. 1985*, pages 327–338. IEEE Comp. Soc. Press, 1985.

[Var99] M. Y. Vardi. Probabilistic linear-time model checking : an overview of the automata-theoretic approach. In *Proc. 5th Int. AMAST Workshop Formal Methods for Real-Time and Probabilistic Systems (ARTS '99), Bamberg, Germany, May 1999*, volume 1601 of *Lecture Notes in Computer Science*, pages 265–276. Springer, 1999.

[ZWR+80] P. Zafiropulo, C. H. West, H. Rudin, D. Cowan, and D. Brand. Towards analysing and synthesizing protocols. *IEEE Transactions on Communication*, 28(4) :651–661, 1980.