UNIVERSITÉ D'AIX-MARSEILLE

ÉCOLE DOCTORALE DE MATHÉMATIQUE ET INFORMATIQUE, E.D. 184

U.F.R SCIENCES

Laboratoire d'Informatique Fondamentale de Marseille, CNRS UMR 7279

THÈSE DE DOCTORAT

Discipline : Informatique

par

Florent AVELLANEDA

sous la direction de Rémi MORIN

Vérification de réseaux de Petri avec états sous une sémantique d'ordres partiels

Soutenue le 10 décembre 2013 devant le jury composé de

Benoît CAILLAUD	IRISA Rennes	Examinateur
Jean-Michel COUVREUR	Université d'Orléans	Examinateur
Stefan HAAR	LSV Cachan	Rapporteur
Hanna KLAUDEL	Université d'Evry	Examinatrice
Jérôme LEROUX	LaBRI Bordeaux	Rapporteur
Rémi MORIN	Université d'Aix-Marseille	Directeur
Peter NIEBERT	Université d'Aix-Marseille	Examinateur

Remerciements

Tout d'abord, je tiens à adresser mes plus sincères remerciements à mon directeur de thèse, Rémi Morin, pour la confiance qu'il m'a accordée en acceptant de diriger cette thèse. Il a été tout au long de ces trois années un encadrant aux grandes qualités pédagogiques, scientifiques et humaines. Je le remercie également pour le temps qu'il m'a consacré. J'ai beaucoup appris à ses côtés et je lui adresse ma plus profonde gratitude.

Je remercie chaleureusement Stefan Haar et Jérôme Leroux, qui m'ont fait l'honneur d'être rapporteurs de ma thèse. Un grand merci également à Benoît Caillaud, Jean-Michel Couvreur, Hanna Klaudel et Peter Niebert qui ont accepté de faire partie du jury.

Je remercie également toutes les personnes que j'ai eu le plaisir de côtoyer pendant ces années et qui m'ont soutenu d'une façon ou d'une autre. Merci à mes compagnons de bureau, présents et passés, pour la bonne ambiance quotidienne dans laquelle j'ai pu accomplir mon travail. Merci aux membres de l'équipe MOVE, qui m'ont accueilli et accompagné durant mon doctorat. Merci également à l'équipe ACRO pour m'avoir permis d'assister à leur groupe de travail et pour tous les conseils avisés qu'ils m'ont prodigués. Merci aux secrétaires pour leurs disponibilités et pour l'aide qu'elles m'ont apportée. Je tiens à remercier plus généralement l'ensemble des membres du laboratoire pour la bonne humeur générale qui y règne. Toutes ces années n'auraient pas été aussi enrichissantes sans l'apport de chacun.

J'adresse finalement un grand merci à ma famille, pour m'avoir soutenu depuis toujours.

Table des matières

Chapitre

1 Introduction

1.1 Contexte

Le XXe siècle a été marqué par l'arrivée de l'électronique, une branche de la physique appliquée, traitant entre autres de la mise en forme et de la gestion de signaux électriques. L'assemblage de divers composants électroniques permet de concevoir des systèmes « intelligents » assurant une fonction spécifique. Le nombre de ces systèmes ne cesse d'augmenter et d'apporter leur contribution pour la résolution de problèmes de plus en plus complexes.

Plusieurs systèmes électroniques peuvent être utilisés conjointement afin d'effectuer des tâches compliquées. Pour simplifier, nous avons recours à un assemblage de plusieurs sous-systèmes travaillant en collaboration ce qui implique la résolution efficace du problème des communications. La communication entre ces sous-systèmes doit être sûre et pertinente. En effet, si l'une des communications entre deux sous-systèmes est défectueuse, alors l'ensemble du système est perturbé et perd de sa fiabilité. Les systèmes doivent non seulement parler un langage commun, mais aussi suivre un ensemble de règles appelé protocole de communication. Afin d'assurer des communications sans erreur entre les différents systèmes, les protocoles doivent être correctement spécifiés.

Avec l'arrivée des ordinateurs et plus spécifiquement d'Internet, les communications entre systèmes deviennent de plus en plus complexes. Il est alors difficile d'établir l'ensemble des règles des protocoles et de certifier que celui-ci est correctement spécifié. Pour cette raison, afin de mieux visualiser les interactions, les protocoles de communication sont généralement spécifiés de manière graphique. De plus, diverses propriétés peuvent être vérifiées automatiquement sur ces représentations graphiques afin de garantir que certains types de défauts sont exclus.

1.2 Contributions

Les MSG (pour « Message Sequence Graphs ») sont un formalisme bien connu et souvent utilisé pour décrire des ensembles de scénarios de manière visuelle dans le domaine des protocoles de communication. Les exécutions du système communicant sont alors représentées par des ordres partiels d'événements répartis sur des sites distants, appelés instances. Nous nous intéressons dans la première partie de la thèse à la détection de la *divergence* [18] et à la vérification de la *coopération globale* [50, 51] pour un MSG donné. Nous considérons également les propriétés d'accessibilité et de couverture relatives au contenu des canaux au cours des exécutions. Notre première contribution consiste à utiliser des solveurs SAT afin de résoudre ces problèmes efficacement. Nous comparons cette approche aux outils existants à l'aide de plusieurs benchmarks. Bien que les MSG aient été l'objet de nombreux travaux dans la littérature, on constate qu'ils pâtissent d'une certaine rigidité. En effet, la notion de variables ou de compteurs est souvent écartée par souci de simplification ce qui

réduit l'expressivité du formalisme. De même les contraintes temporelles sont exclues car elle conduisent à l'indécidabilité de propriétés de base telles que la divergence [28]. Nous étudions donc dans la suite de cette thèse un modèle plus général permettant de vérifier le même type de propriétés pour des spécifications plus complexes.

Considérons un ensemble de réactions qui ont lieu au sein d'une collection de particules telles que chaque réaction consomme un multi-ensemble de particules disponibles et produit une combinaison linéaire d'autres types de particules. Considérons de plus un état de contrôle qui détermine si une règle peut se produire ou non et tel que l'exécution de cette règle conduit à un nouvel état de contrôle. Ce modèle correspond aux réseaux de Petri avec états (ou PNS pour « Petri Net with States ») que nous introduisons dans la seconde partie de la thèse. Les PNS sont simplement des systèmes d'addition de vecteurs avec états sans la contrainte de se limiter à des règles pures. Nous présentons une sémantique d'ordres partiels pour les PNS qui étend la sémantique habituelle des processus des réseaux de Petri. L'approche est simple et naturelle. D'abord, nous considérons l'ensemble des séquences de calculs franchissables d'un PNS ; ensuite nous définissons les processus qui représentent une séquence donnée. Ainsi, chaque processus décrit des dépendances causales entre des événements partiellement ordonnés. Cela signifie que deux règles qui apparaissent l'une après l'autre dans une séquence de règles peuvent se produire simultanément dans un processus. Le modèle des MSG étudié dans la première partie peut alors être considéré comme un cas particulier de PNS où les réactions autorisées sont uniquement l'émission et la réception de messages entre des instances. Nous expliquons comment modéliser des compteurs ou des timers dans des MSG étendus sous la forme de PNS. Étendre les MSG dans le cadre des réseaux de Petri est une idée naturelle déjà explorée sous d'autres formes dans la littérature [12, 16, 17, 29].

Nous nous intéressons à trois problèmes de vérification classiques sur l'ensemble des marquages accessibles par les préfixes des processus : le caractère borné, la couverture et l'accessibilité. Nous montrons comment réduire ces problèmes au cas particulier des réseaux de Petri de telle sorte que tous les résultats de complexité et de décidabilité s'étendent des réseaux de Petri aux PNS sous la sémantique des processus. Nous montrons également que ces résultats s'appliquent au cas particulier des MSG étendus sous la restriction FIFO. Nous obtenons ainsi un modèle qui étend les MSG à l'aide de compteurs et de timers et pour lequel les propriétés élémentaires telles que le caractère borné ou l'accessibilité sont décidables, contrairement au formalisme étudié dans [48]. Nous expliquerons cette contradiction apparente en comparant brièvement les deux approches du point de vue de la spécification de contraintes temporelles et de la sémantique adoptée.

La notion de non-divergence pour un MSG coïncide avec le caractère borné par préfixe pour les PNS. Cependant, l'expressivité des PNS est plus grande que celle des MSG. En particulier, la non-divergence d'un MSG est coNP-complet alors que le caractère borné par préfixe d'un PNS requiert un espace exponentiel. Pour contourner ce problème, nous proposons dans la troisième partie de la thèse d'abstraire les propriétés à vérifier, de sorte que si l'abstraction d'une propriété est satisfaite, alors cette propriété est satisfaite. L'abstraction que nous proposons consiste à vérifier les propriétés de manière structurelle, c'est-à-dire à vérifier si, quel que soit le marquage initial, la propriété reste satisfaite. Nous pouvons citer

par exemple le caractère structurellement borné qui assure que, quel que soit le marquage initial, le nombre de particules est borné au cours des exécutions. Cette abstraction est intéressante, car vérifier le caractère borné d'un réseau de Petri fourni avec une configuration initiale nécessite un espace exponentiel [43, 74] alors que vérifier le caractère structurellement borné est polynomial [43, 81]. Nous observons que la vérification d'une propriété structurelle pour un PNS se résume à trouver un cycle particulier, que nous appelons cycle pathologique, dans un graphe. Dans [68], Kosaraju et Sullivan ont montré comment vérifier l'existence de tels cycles en temps polynomial.

Nous introduisons la notion de propriété semi-structurelle afin de considérer des PNS paramétrés. Cela consiste à fixer le marquage initial d'un sous-ensemble approprié de places, puis à vérifier les propriétés structurelles sur les places restantes. Nous pouvons ainsi vérifier une propriété sans fixer la valeur initiale de certaines variables vue comme des paramètres du système. Néanmoins le degré d'abstraction adopté est parfois excessif et la propriété structurelle obtenue insatisfaite. Il est alors utile d'analyser le bug détecté afin de moduler l'abstraction.

Une caractéristique particulièrement attrayante des MSG et des PNS réside dans leur représentation graphique similaire à un automate. Il est donc intéressant de décrire les bugs de manière visuelle. Toutefois, la longueur minimale d'un cycle pathologique dans un PNS peut être exponentielle en la taille du PNS. Par conséquent, l'énumération de la séquence des arcs décrivant un bug est prohibitive et nous avons besoin de concevoir une représentation compacte des cycles pathologiques. Pour cela, nous introduisons la notion de *lasso*. Schématiquement, un lasso comprend un cycle muni de deux chemins aller et retour vers un nœud initial fixé. En outre, la *valuation* d'un lasso détermine le nombre d'itérations de la composante cyclique. Nous montrons comment calculer en temps polynomial un multi-ensemble de lassos élémentaires avec un nœud de départ commun qui correspond à un multi-ensemble d'arcs représentant un cycle pathologique. Il est intéressant de noter que le nombre de lassos élémentaires distincts dont nous avons besoin est au plus égal à la dimension des vecteurs de poids des règles du PNS. De plus, le point de départ commun de ces lassos peut être choisi arbitrairement.

Trouver les contre-exemples les plus courts possible est souvent souhaitable en pratique pour comprendre le mauvais comportement d'un système. Nous observons que la plupart des problèmes de minimisation naturels sont NP-difficiles. Nous montrons ensuite comment minimiser la taille des lassos des contre-exemples en temps polynomial. Ce résultat s'appuie sur les travaux fondamentaux de Grötschel, Lovász et Schrijver [97] qui montrent comment résoudre dans certains cas en temps polynomial un système d'inéquations ayant un nombre exponentiel de contraintes.

1.3 Plan de la thèse

Cette thèse se compose de trois parties distinctes. La première partie est consacrée aux MSG. La second partie est consacrée aux réseaux de Petri avec états. La dernière partie est consacrée à la vérification de propriétés structurelles des réseaux de Petri avec états. Voici

le plan chapitre par chapitre.

Chapitre 2 : Toutes les notions et les notations utilisées dans au moins deux des trois parties de cette thèse sont rassemblées dans ce chapitre.

Partie I

Chapitre 3 : Nous présentons la problématique de la première partie et introduisons les notions essentielles qu'elle utilise telles que les MSC et les MSG.

Chapitre 4 : Nous présentons les propriétés que nous étudions sur les MSG : l'accessibilité, la couverture, la divergence et la coopération globale. Chacune de ces propriétés correspond à un problème NP-complet.

Chapitre 5 : Nous modélisons chacune de ces propriétés par des formules booléennes dans le but de les vérifier efficacement à l'aide de solveurs SAT.

Chapitre 6 : Nous mesurons expérimentalement l'efficacité de cette approche. Pour cela, nous comparons nos résultats à divers outils existants à l'aide d'un prototype.

Partie II

Chapitre 7 : Nous présentons la problématique de la seconde partie et introduisons la sémantique d'ordres partiels adoptée pour les PNS. Nous montrons comment les processus sont créés et comment ce formalisme généralise le formalisme des MSG.

Chapitre 8 : Nous montrons comment nous pouvons vérifier une formule de la logique MSO sur l'ensemble des processus d'un PNS borné.

Chapitre 9 : Nous montrons comment les propriétés d'accessibilité par préfixe peuvent être vérifiées à l'aide d'une réduction aux réseaux de Petri.

Partie III

Chapitre 10 : Nous présentons la problématique des propriétés structurelles et introduisons également la notion de propriétés semi-structurelles. Nous expliquons pourquoi vérifier ces propriétés revient à détecter des cycles pathologiques.

Chapitre 11 : Nous rappelons comment on peut détecter, sous la forme d'un multi-ensemble d'arcs, des cycles pathologiques pour le caractère structurellement borné et la terminaison structurelle en temps polynomial à l'aide de la programmation linéaire.

Chapitre 12 : Afin de représenter des cycles pathologiques de manière simple et compacte, nous introduisons la notion de lasso. Nous montrons dans ce chapitre comment calculer en temps polynomial un multi-ensemble de lassos élémentaires avec un nœud de départ commun qui correspond à un multi-ensemble donné d'arcs représentant un cycle pathologique.

Chapitre 13 : Trouver les contre-exemples les plus courts est souvent souhaitable en pratique pour comprendre le mauvais comportement d'un système. Nous montrons dans ce chapitre comment minimiser la taille des lassos des contre-exemples en temps polynomial.

Chapitre

2 Préliminaires

Ce chapitre introduit les notions et les notations utiles dans au moins deux des trois parties de cette thèse. Nous commençons par introduire les graphes qui sont utilisés tout au long de cette thèse. Vient ensuite la notion d'ordres partiels qui servira pour les deux premières parties. Nous terminons en définissant les réseaux de Petri, les VAS et les VASS qui seront utilisés pour les deux dernières parties.

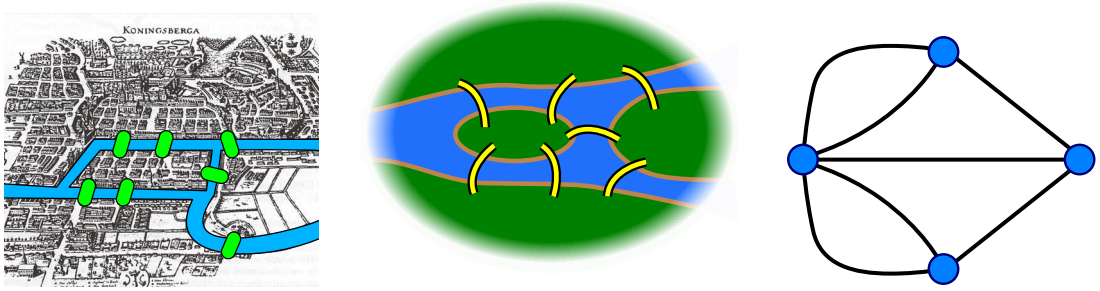
2.1 Graphe

Un *graphe* est un objet mathématique composé d'un ensemble de points appelés *sommets* ou *nœuds* et d'un ensemble de liens reliant certaines paires de nœuds entre eux. Ces liens peuvent être orientés, on les appelle alors des *arcs*, ou non orienté, on les appelle alors des *arêtes*.

Les graphes sont utilisés dans beaucoup de domaines afin de modéliser et abstraire des problèmes. La théorie des graphes a été introduite initialement par Euler dans [45] avec le problème des sept ponts de Königsberg.

Exemple 2.1.1 (Wikipedia)**.** *La ville de Königsberg (aujourd'hui Kaliningrad) est construite autour de deux îles situées sur le Pregel et reliées entre elles par un pont. Six autres ponts relient les rives de la rivière à l'une ou l'autre des deux îles, comme représentés sur la figure 2.1(a). Le problème consiste à déterminer s'il existe ou non une promenade dans les rues de Königsberg permettant, à partir d'un point de départ au choix, de passer une et une seule fois par chaque pont, et de revenir à son point de départ, étant entendu qu'on ne peut traverser le Pregel qu'en passant sur les ponts.*

Afin de résoudre ce problème, Euler représente la ville sous la forme d'un graphe (figure 2.1(c)). On obtient ainsi une représentation simple et concise du problème.



(a) (b) (c)

Figure 2.1 – Problème des sept ponts de Königsberg.

La représentation sous forme de graphe a plusieurs avantages. En plus d'être une représentation simple, beaucoup de problèmes peuvent être modélisés avec des graphes. Ainsi tous

les résultats théoriques existant sur les graphes servent pour un grand nombre de domaines.

Par exemple Euler caractérise une classe de graphes dits eulériens.

Définition 2.1.2 (Graphe eulérien)**.** *Un graphe est dit* eulérien *s'il existe un cycle passant une et une seule fois par chaque arc du graphe.*

Il montre que cette classe de graphes se caractérise simplement par le théorème suivant.

Théorème 2.1.3 (Théorème d'Euler)**.** *Un graphe connexe est eulérien si et seulement si chacun de ses sommets est incident à un nombre pair d'arêtes.*

Notons que ce théorème fait référence à un graphe non orienté. Nous verrons plus tard le cas orienté avec la notion de multi-ensemble d'arcs Eulérien (définition 11.1.1).

Ainsi, le problème 2.1.1 se résout facilement en utilisant ce théorème. Comme le graphe de la figure 2.1(c) a trois nœuds incidents à un nombre impair d'arêtes, on peut en déduire qu'il n'existe pas de cycle passant une et une seule fois par chaque nœud du graphe.

Définition formelle et notations

Formellement, un graphe G est un couple $G = (V, A)$ avec V un ensemble de nœuds et A un ensemble d'arcs. Chaque arc $a \in A$ relie un nœud q_1 à un nœud q_2 ; les deux nœuds q_1 et q_2 sont appelés respectivement le *domaine* et le *codomaine* de a et sont notés $dom(a)$ et $cod(a)$ respectivement.

Définition 2.1.4 (séquence d'arcs)**.** *Un chemin est une séquence d'arcs $\gamma = a_1...a_n \in A^\star$ tel que $dom(a_{i+1}) = cod(a_i)$ pour chaque $i \in [1..n-1]$.*

Définition 2.1.5 (chemin fermé et cycle)**.** *Un chemin $\gamma = a_1...a_n \in A^\star$ est fermé si $n \geq 1$ et $dom(a_1) = cod(a_n)$. Un chemin fermé est appelé un* cycle*.*

Définition 2.1.6 (chemin élémentaire)**.** *Un chemin $\gamma = a_1...a_n \in A^\star$ est* élémentaire *s'il ne passe pas deux fois par un même nœud, c'est-à-dire $dom(a_i) \neq dom(a_j)$ pour tous $i < j$.*

Définition 2.1.7 (cycle élémentaire)**.** *Un cycle $\gamma = a_1...a_n \in A^\star$ est* élémentaire *si γ est un chemin élémentaire.*

Définition 2.1.8 (chaîne)**.** *Une chaîne reliant x à y est définie par une suite finie d'arêtes consécutives, reliant x à y.*

Définition 2.1.9 (nœuds connexes)**.** *Deux nœuds sont dit connexes s'il existe une chaîne entre ces deux nœuds.*

Définition 2.1.10 (nœuds fortement connexes)**.** *Deux nœuds sont dit fortement connexes s'il existe un chemin d'aller et de retour entre ces deux nœuds.*

Définition 2.1.11 (graphe connexe)**.** *Un graphe est dit connexe si toute paire de nœuds est connexe.*

Définition 2.1.12 (graphe fortement connexe)**.** *Un graphe est dit fortement connexe si toute paire de nœuds est fortement connexe.*

2.2 Relation d'ordre et ordre partiel

L'utilisation d'une relation d'ordre est une chose qui nous est très naturelle. En effet, dès notre plus jeune age, les bébés sont capables de dénombrer jusqu'à 4 éléments sans aucun apprentissage préalable. Ainsi, ils comprennent que « trois bonbons » est supérieur à « deux bonbons ». Cette notion est ensuite approfondie à l'école dans les cours de mathématiques : $5 > 2$, $2 = 2$, $2 < 3$, etc. La notion de supériorité ou d'infériorité numéraire est l'une des intuitions fondamentales des systèmes numériques.

Si nous souhaitons maintenant comparer des ensembles d'éléments, cette relation d'ordre devient partielle. En effet, si un ensemble A contient tous les éléments d'un ensemble B, alors B est inférieur ou égal à A. Par contre, si A contient des éléments non contenus dans B et B contient des éléments non contenus dans A, alors A et B ne sont pas comparables.

Suivant une approche classique en théorie de la concurrence [71, 54, 89, 37] les exécutions d'un système parallèle ou distribué seront représentées dans cette thèse par des ordres partiels étiquetés, également appelés multi-ensembles partiellement ordonnés ou mots partiels.

Définition formelle et notations

Un *mot partiel* sur un alphabet Σ est un triplet $(E, \preceq, \xi)$ où $(E, \preceq)$ est un ordre partiel fini et ξ est une application de E vers Σ. Un mot partiel peut être considéré comme une abstraction de l'exécution d'un système concurrent. Les éléments de E sont des *événements* et la lettre $\xi(e)$ décrit l'action de base qui est effectuée par l'événement $e \in E$. Par ailleurs, l'ordre $\preceq$ décrit la dépendance causale entre les événements. D'un point de vue mathématique, nous ne distinguerons pas les mots partiels isomorphes. Une *extension linéaire* d'un mot partiel $t = (E, \preceq, \xi)$ est un ordre total $\leq$ sur E qui respecte l'ordre causal, c'est-à-dire $\preceq \subseteq \leq$. Elle correspond intuitivement à une vue séquentielle de l'exécution t et peut naturellement être considérée comme un mot sur Σ. Par $\mathrm{LE}(t) \subseteq \Sigma^*$, nous désignons l'ensemble des extensions linéaires de t. Pour tous les couples d'événements $e, f \in E$ nous écrivons $e \prec\!\!\!\cdot\, f$ si $e \prec f$ et $e \prec f' \preceq f$ implique $f' = f$. Cela signifie simplement que e est avant f dans le diagramme de Hasse de $(E, \preceq)$. Pour chaque mot partiel $M = (E, \preceq, \xi)$ sur Σ, nous notons $\#^a(M)$ le nombre d'événements marqués par $a \in \Sigma$ dans M. En outre, pour chaque événement $e \in E$, $\downarrow_M e$ désigne la restriction de M aux événements précédant e, c'est-à-dire le préfixe minimal de M qui contient e.

2.3 Réseaux de Petri

Les réseaux de Petri ont été introduits en 1962 par Carl Adam Petri [87]. Possédant une représentation graphique simple, les réseaux de Petri permettent de modéliser des systèmes dynamiques échangeant des ressources. La notion de *places* peut représenter différents types de composants au sein d'un système : un état local d'un processus séquentiel, un canal de communications, un registre partagé, un type de particule, une molécule dans un système chimique, etc. Un multi-ensemble de places est habituellement appelé un *marquage* et il est considéré comme une distribution de *jetons* dans les places.

Les réseaux de Petri sont des graphes orientés composés d'un nombre fini de nœuds places et d'un nombre fini de nœuds transitions. Des arcs orientés peuvent relier des nœuds places à des nœuds transitions ou des nœuds transitions à des nœuds places.

Les nœuds places, représentés par des cercles, représentent des ressources disponibles. Le nombre de ressources disponibles est représenté par des points (appelés jetons) à l'intérieur de ces nœuds places. Un jeton représente donc une ressource.

Les nœuds transitions, représentés par des rectangles pleins, représentent des actions pouvant effectuer des échanges de jetons entre différentes places. L'orientation des arcs indique les échanges pouvant être effectués. Chaque échange consomme un jeton dans toutes les places pointant vers une transition et produit un jeton dans les places pointées par cette transition.

Exemple 2.3.1. *Voici un exemple de réseau de Petri (figure 2.2).*

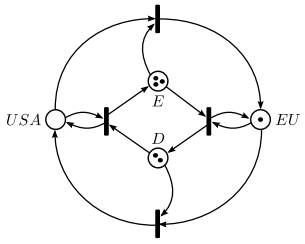


FIGURE 2.2 – Échangeur de monnaie avec un réseau de Petri.

Ce réseau de Petri modélise un échangeur de monnaie. Les places E et D représentent respectivement une quantité d'euros et une quantité de dollars. Lorsqu'un jeton est présent dans la place EU, les euros peuvent être échangés contre des dollars, et lorsqu'un jeton est présent dans la place USA, les dollars peuvent être échangés contre des euros. Cependant, le déplacement d'un jeton de EU vers USA (ou inversement) coûte de l'argent.

En général, le nombre de jetons consommés n'est pas égal au nombre de jetons produits. Notons également que si une place ne contient pas de jeton, alors on ne peut exécuter (on dit aussi « tirer ») aucune des transitions pointées par cette place.

Définition formelle et notations

Définition 2.3.2 (Réseau de Petri)**.** *Un réseau de Petri est un quadruplet $\mathcal{N} = (P, T, W, \mu_{in})$ où :*

- *P est un ensemble fini de places et T est un ensemble fini de transitions telles que $P \cap T = \emptyset$.*
- *W est une application de $(P \times T) \cup (T \times P)$ vers $\mathbb{N}$.*
- *μ_{in} est une application de P vers $\mathbb{N}$ appelé le marquage initial.*

Nous identifions l'état dans lequel se trouve un système par son *marquage*. Cela correspond au nombre de jetons présents dans chacune de ses places, c'est-à-dire une application $\mu : P \to \mathbb{N}$, autrement dit un *multi-ensemble* d'éléments de P. Soit $\mu \in \mathbb{N}^P$ un marquage

et $p \in P$ une place, alors $\mu(p)$ est le nombre de jetons contenus dans la place p pour le marquage μ.

Afin de simplifier l'utilisation des transitions, nous rappelons les notions et les notations classiques de preset et postset.

Définition 2.3.3 (Preset, postset)**.** *Le preset d'une transition $t \in T$, noté ${}^\bullet t$, correspond au multi-ensemble de places pointant vers t : ${}^\bullet t = \sum_{p \in P} W(p,t) \cdot p$. Le postset d'une transition $t \in T$, noté $t^\bullet$, correspond au multi-ensemble de places pointées par t : $t^\bullet = \sum_{p \in P} W(t,p) \cdot p$.*

Intuitivement, le preset correspond aux jetons dont une transition a besoin pour être appliquée, et postset correspond aux jetons produits par cette règle. Ainsi, on dit qu'une transition t est *tirable* à partir d'un marquage μ si $\mu(p) \geq {}^\bullet t(p)$ pour chaque place $p \in P$, ce que l'on note $\mu \geq {}^\bullet t$. Nous pouvons généraliser cette notion de transition tirable aux séquences de transitions. On dit qu'une séquence de transitions $s = t_1 ... t_n \in T^*$ est *tirable* à partir d'un marquage μ s'il y a des marquages μ_0, ..., μ_n tels que $\mu_0 = \mu$ et pour chaque $k \in [1, n]$: $\mu_{k-1} \geq {}^\bullet t_k$ et $\mu_k = \mu_{k-1} - {}^\bullet t_k + t_k^\bullet$. Cela signifie que les transitions de s peuvent être appliquées dans l'ordre à partir du marquage μ. Chaque transition t_k consomme ${}^\bullet t_k$ jetons de μ_{k-1} et produit $t_k^\bullet$ nouveaux jetons dans le marquage μ_k. On dit alors que μ_n est *atteint* par la séquence de règle s à partir du marquage μ. Nous pouvons également dire que s conduit au marquage μ_s.

Voici quelques propriétés classiques pour les réseaux de Petri.

Définition 2.3.4 (Accessibilité)**.** *Un marquage est* accessible *dans un réseau de Petri $\mathcal{N}$ s'il est atteint par une séquence de transitions tirable à partir du marquage initial.*

L'accessibilité dans un réseau de Petri (et dans d'autres formalismes) est un problème classique en vérification. En effet, savoir si une configuration donnée est accessible peut être très utile en pratique. Si un bug spécifique peut être identifié par un marquage précis, alors l'accessibilité permet de savoir si un tel bug peut avoir lieu. Ce problème est connu pour être décidable et EXPSPACE-difficile pour les réseaux de Petri [74, 94, 80, 67, 70, 72, 73].

Définition 2.3.5 (Caractère borné)**.** *Un réseau de Petri est dit* borné *si l'ensemble de ses marquages accessibles est fini.*

Le caractère borné d'un réseau de Petri est très intéressant. Les places représentant généralement des types de ressources, avoir la garantie qu'il existe une borne pour chaque ressource est utile en pratique. Ce problème est EXPSPACE-complet pour les réseaux de Petri [74, 90].

Définition 2.3.6 (Couverture)**.** *Un marquage μ est couvert s'il existe un marquage accessible μ' tel que $\mu \leq \mu'$.*

Le problème de la couverture d'un réseau de Petri est aussi EXPSPACE-complet [74, 90].

Définition 2.3.7 (Terminaison)**.** *Un réseau de Petri est dit* terminant *si l'ensemble des séquences de transitions tirables, depuis le marquage initial, est fini.*

Bien que l'on cherche généralement à éviter les blocages dans les systèmes concurrents, la terminaison reste dans certains cas, un problème de base de la vérification formelle : en particulier, la non-terminaison peut résulter d'un *livelock* pour un programme concurrent lorsque les composants ne parviennent pas à exécuter leurs tâches.

2.4 Réseau causal

Si la fonction de poids W ne prend que des valeurs binaires, alors elle est souvent décrite comme une relation $F \subseteq (P \times T) \cup (T \times P)$ où $(x, y) \in F$ si $W(x, y) = 1$. Alors, F^+ désigne la fermeture transitive de F.

Définition 2.4.1. *[42, 103] Un* réseau causal *est un réseau de Petri $\mathcal{K} = (B, E, F, \mu_{min})$ dont les places sont appelées des* conditions*, les transitions sont appelées des* événements*, et la fonction de poids prend ses valeurs dans $\{0, 1\}$ et se représente par une relation $F \subseteq (B \times E) \cup (E \times B)$ qui satisfait les exigences suivantes :*

1. *le graphe est acyclique, c'est-à-dire pour tout $x, y \in B \cup E$, $(x, y) \in F^+$ implique $(y, x) \notin F^+$.*
2. *les conditions ne sont pas « branchées », c'est-à-dire $|{}^\bullet b| \leq 1$ et $|b^\bullet| \leq 1$ pour tout $b \in B$.*
3. *les conditions minimales correspondent au marquage initial : pour tout $b \in B$, $\mu_{min}(b) = 1$ si ${}^\bullet b = \emptyset$ et $\mu_{min}(b) = 0$ sinon.*

Notons que la troisième condition garantit que le marquage initial μ_{min} peut être récupéré à partir de la structure (B, E, F), car elle coïncide avec l'ensemble des conditions minimales. Pour cette raison, les réseaux causaux sont souvent définis comme un triplet (B, E, F) satisfaisant les deux premières conditions de la définition 2.4.1. Dans la littérature, les réseaux causaux sont aussi appelés *réseaux d'occurrences* [20, 52, 54, 53, 91]. Cependant, des réseaux de Petri plus généraux sont aussi appelés réseaux d'occurrences dans le domaine des dépliages partiels ou des processus branchants [42, 44, 85].

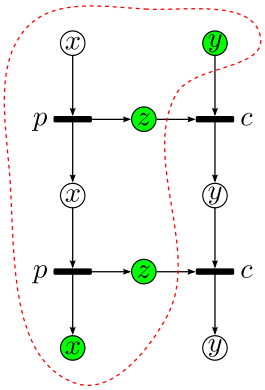


FIGURE 2.3 – Préfixe d'un réseau causal.

La fermeture transitive et réflexive F^* de la relation F dans un réseau causal $\mathcal{K} = (B, E, F, \mu_{min})$ donne un ordre partiel sur l'ensemble d'événements E. Une *configuration* est

un sous-ensemble d'événements $H \subseteq E$ qui est fermé vers le bas, c'est-à-dire que $e'F^*e$ et $e \in H$ implique $e' \in H$. Chaque configuration H détermine un *réseau causal préfixe* $\mathcal{K}_H$ dont les événements sont précisément les événements de H et dont les places se composent des places minimales de $\mathcal{K}$ (par rapport à la relation d'ordre partiel F^*) et de toutes les places liées à un événement de H. La figure 2.3 illustre un sous-ensemble d'un réseau causal (encerclé par une ligne pointillée) qui est un préfixe de ce réseau causal. Pour chaque classe de réseaux causaux $\mathcal{L}$, nous notons $\mathrm{Pref}(\mathcal{L})$ la classe de tous les préfixes de tous les réseaux causaux de $\mathcal{L}$.

2.5 Réseaux de Petri avec états

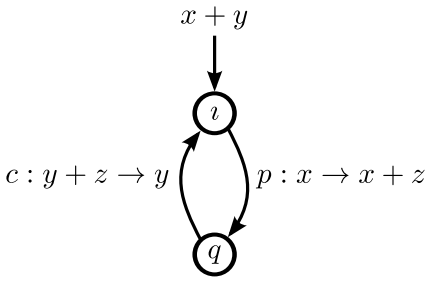
Le cadre général de cette thèse est formalisé par le modèle des réseaux de Petri avec états. Il s'agit simplement de munir les réseaux de Petri d'un état de contrôle qui inhibe certaines transitions. De plus l'exécution d'une transition permet de changer d'état et donc de modifier l'ensemble des inhibitions. Nous fixons tout d'abord un ensemble fini de places P et un ensemble fini de noms de règles N.

Une règle est un moyen de produire de nouveaux jetons dans certaines places en consommant des jetons d'autres places. Formellement une *règle* est un triplet $r = (\lambda, \alpha, \beta)$ où $\lambda \in N$ est un nom de règle et $\alpha, \beta \in \mathbb{N}^P$ sont des marquages appelés *garde* et la *mise-à-jour*, respectivement. Une telle règle est notée par $\lambda : \alpha \to \beta$. Cela signifie intuitivement qu'un multi-ensemble de jetons α peut être consommé pour produire un multi-ensemble de jetons β de façon atomique. Des règles différentes peuvent partager la même garde α et la même mise-à-jour β. C'est pourquoi des noms de règle sont utilisés afin de distinguer les règles similaires, mais distinctes. Pour chaque règle $r = (\lambda, \alpha, \beta)$, nous notons ${}^\bullet r = \alpha$ et $r^\bullet = \beta$.

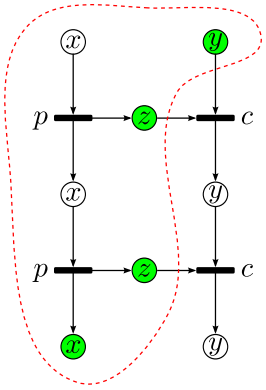
Définition 2.5.1. *Un* réseau de Petri avec états *(ou* PNS *pour « Petri Net with States »)* *sur un ensemble de règles R est un automate $\mathcal{S} = (Q, \imath, \longrightarrow, \mu_{in})$ où Q est un ensemble fini d'états, $\imath$ est un état initial, $\longrightarrow \in Q \times R \times Q$ est un ensemble fini d'arcs étiquetés par des règles, et $\mu_{in} \in \mathbb{N}^P$ est un marquage initial.*

Soit $\mathcal{S} = (Q, \imath, \longrightarrow, \mu_{in})$ un PNS. Un arc étiqueté $(q_1, r, q_2) \in \longrightarrow$ sera noté par $q_1 \xrightarrow{r} q_2$. Une séquence de règles $s = r_1...r_n \in R^*$ est appelée une *séquence de règles de $\mathcal{S}$* s'il y a des états $q_0, ..., q_n \in Q$ tels que $\imath = q_0$ et pour chaque $i \in [1, n]$, un arc $q_{i-1} \xrightarrow{r_i} q_i$. Ces conditions seront notées $\imath \xrightarrow{s} q_n$. Par exemple, $(p : x \to x + z) \cdot (c : y + z \to y) \cdot (p : x \to x + z) \cdot (c : y + z \to y)$ est une séquence de règles du PNS à deux états représenté sur la figure 2.4(a). On note $CS(\mathcal{S})$ l'ensemble de toutes les séquences de règles de $\mathcal{S}$. Ce langage de mots de R^* est évidemment régulier et clos par préfixe. Inversement tout langage régulier et clos par préfixe correspond à l'ensemble des séquences de règles d'un PNS.

Une séquence de règles $s = r_1...r_n \in R^*$ est *tirable* à partir d'un marquage μ s'il y a des multi-ensembles de places μ_0, ..., μ_n tels que $\mu_0 = \mu$ et pour chaque $k \in [1, n]$: $\mu_{k-1} \geq {}^\bullet r_k$ et $\mu_k = \mu_{k-1} - {}^\bullet r_k + r_k^\bullet$. Cela signifie intuitivement que chaque règle de s peut être appliquée à partir du marquage μ dans l'ordre linéaire spécifié par s. Chaque règle r_k consomme ${}^\bullet r_k$ jetons de μ_{k-1} et produit $r_k^\bullet$ nouveaux jetons qui donnent le marquage μ_k. On dit alors



(a) Un PNS avec deux états de contrôle

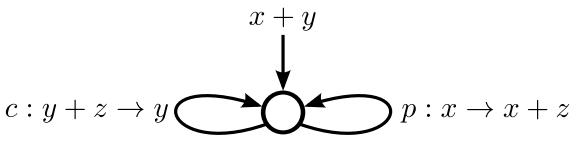


(b) Un processus et un préfixe

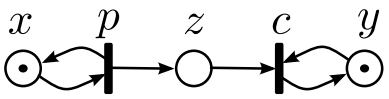
FIGURE 2.4 – Exemple du producteur consommateur.

que μ_n est atteignable par la séquence de règles s à partir du marquage μ. Nous pouvons également dire que s conduit au marquage μ_n. On note par $FCS(\mathcal{S})$ l'ensemble de toutes les séquences de règles de $\mathcal{S}$ tirables. Un marquage est dit *accessible* dans $\mathcal{S}$ s'il est atteint par une séquence de règles de $\mathcal{S}$ tirable. Un PNS est dit *borné* si l'ensemble de ses marquages accessibles est fini.

Expliquons à présent pourquoi les réseaux de Petri peuvent être identifiés comme des PNS particuliers munis d'un unique état. Soit $\mathcal{N} = (P, T, W, \mu_{in})$ un réseau de Petri. Nous allons considérer $\mathcal{N}$ comme un PNS $\mathcal{S}_\mathcal{N}$ avec le même ensemble de places P et le même marquage initial. De plus, $\mathcal{S}_\mathcal{N}$ est muni d'un seul état $\imath$ tel que chaque transition $t \in T$ est représentée par un arc étiqueté formant une boucle $\imath \xrightarrow{r} \imath$ où $r = (t, {}^\bullet t, t^\bullet)$. De cette façon, la classe des réseaux de Petri est fidèlement incluse dans la sous-classe des PNS possédant un seul état et tels que chaque transition possède un nom de règle distinct. Inversement, prenons un PNS $\mathcal{S}$ avec un seul état $\imath$ tel que chaque transition comporte un nom de règle distinct. Le réseau de Petri $\mathcal{N}_\mathcal{S}$ correspondant partage avec $\mathcal{S}$ l'ensemble de ses places et son marquage initial. De plus, chaque boucle $\imath \xrightarrow{r} \imath$ détermine une transition t_r de $\mathcal{N}_\mathcal{S}$ telle que ${}^\bullet t_r = {}^\bullet r$ et $t_r^\bullet = r^\bullet$. Par exemple, le PNS de la figure 2.5(a) correspond au réseau de Petri de la figure 2.5(b).



(a) Un PNS avec un seul état



(b) et le réseau de Petri correspondant

FIGURE 2.5 – Exemple du producteur consommateur.

2.6 Systèmes d'addition de vecteurs

Initialement introduite dans [62], la notion de système d'addition de vecteurs (ou VASS pour « Vector Addition System with States ») peut être formellement définie de plusieurs manières légèrement différentes. Dans cette thèse, un *VASS* est tout simplement un PNS tel que chaque règle r étiquetant un arc est *pure*.

Définition 2.6.1 (règle pure)**.** *Une* règle pure *est une règle r telle que pour toute place $p \in P$, ${}^\bullet r(p) \times r^\bullet(p) = 0$.*

Ainsi, une règle est pure si son preset et son postset ne partagent aucune place. Pour cette raison, chaque règle r d'un VASS peut être représentée par le vecteur $v \in \mathbb{Z}^P$ tel que $v(p) = r^\bullet(p) - {}^\bullet r(p)$ pour chaque place $p \in P$. On pourrait aussi exiger qu'un VASS utilise *un nom de règle unique*, c'est-à-dire que pour toutes les règles $r_1, r_2 \in R$, $r_1^\bullet - {}^\bullet r_1 = r_2^\bullet - {}^\bullet r_2$ implique que $r_1 = r_2$. Ainsi, deux règles similaires devraient porter le même nom de règle. Cette restriction n'a pas d'effet sur les résultats présentés dans cette thèse.

2.7 Panorama des modèles étudiés

Ainsi, nous introduisons le modèle des *réseaux de Petri avec états* est comme un cadre formel minimal qui inclut à la fois les réseaux de Petri et les VASS [14] : les réseaux de Petri sont considérés comme des PNS pourvus d'un seul état alors que les VASS sont tout simplement des PNS utilisant seulement des règles pures. Les réseaux de Petri avec uniquement des règles pures sont appelés des VAS (pour « Vector Addition System »).

Nous étudions dans la première partie de cette thèse le modèle bien connu des Message Sequence Graphs (MSG) que l'on identifiera à une sous-classe de PNS. Comme les règles utilisées dans un MSG ne sont pas pures, les MSG ne sont pas formellement un cas particulier de VASS. Nous introduirons également la classe des *MSG étendus* comme une sous-classe des PNS qui permet d'introduire des variables et des timers dans la spécification d'un protocole. La figure 2.6 décrit les différents modèles considérés dans cette thèse.

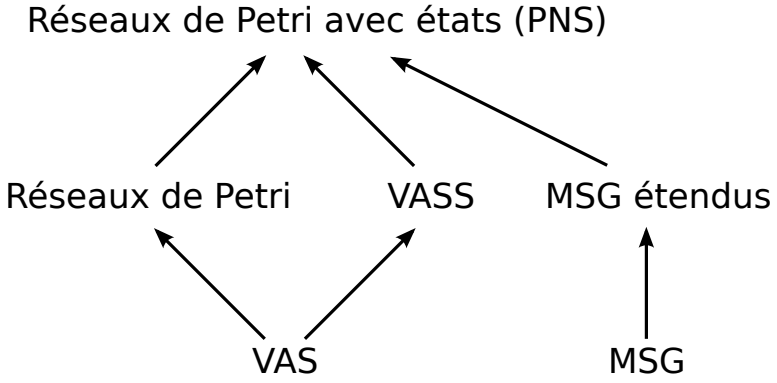


FIGURE 2.6 – Panorama des différents modèles étudiés.

Simulation d'un PNS par un réseau de Petri

Rappelons comment un VASS k-dimensionnel ou, plus généralement, un PNS $\mathcal{S}$ avec k places peut être simulé par un réseau de Petri $\mathcal{N}$ avec $k+n$ places, où n est le nombre d'états [93, 86]. La construction habituelle est illustrée par la figure 2.7 qui montre sur le côté gauche un PNS $\mathcal{S}$ avec 2 états ($\imath$ et q) et 3 places (x, y et z) et sur le côté droit, le réseau de Petri $\mathcal{N}$ correspondant avec 5 places : chaque place de $\mathcal{S}$ et chaque état de $\mathcal{S}$ correspond à une place de $\mathcal{N}$. Le marquage initial de $\mathcal{N}$ décrit le marquage initial de $\mathcal{S}$ et un jeton est ajouté dans la place correspondant à l'état initial de $\mathcal{S}$. En outre, chaque arc $q_1 \xrightarrow{r} q_2$ dans $\mathcal{S}$ est représenté par une transition dans $\mathcal{N}$. Il est facile de voir qu'il y a une correspondance une à une entre les séquences de règles tirables de $\mathcal{S}$ et les séquences de règles tirables de $\mathcal{N}$. De plus, un marquage atteint par $\mathcal{N}$ décrit un marquage atteignable par $\mathcal{S}$ ainsi que son état courant.

Cette construction de $\mathcal{N}$ à partir de $\mathcal{S}$ est intéressante, car elle permet d'analyser l'ensemble des marquages accessibles par $\mathcal{S}$ en utilisant les techniques issues de la littérature des réseaux de Petri [43]. En particulier, le problème du caractère borné demande si l'ensemble des marquages accessibles d'un PNS est fini alors que le problème de couverture demande si un marquage μ donné peut être couvert par un marquage accessible μ', c'est-à-dire $\mu \leq \mu'$. Ces deux problèmes sont décidables (pour les réseaux de Petri et donc pour les réseaux de Petri avec états), mais peuvent nécessiter un espace exponentiel [90].

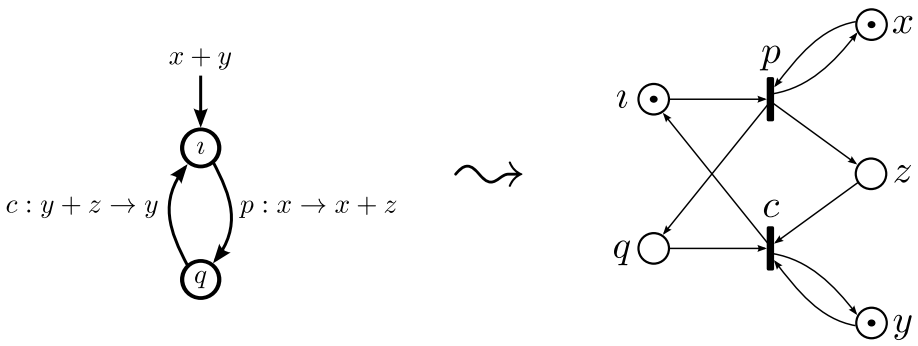


FIGURE 2.7 – Simulation d'un PNS par un réseau de Petri.

La simulation d'un PNS par un réseau de Petri nous amène au résultat suivant.

Propriété 2.7.1. *Soit $\mathcal{S}$ un PNS et r une règle attachée à un arc de $\mathcal{S}$. Nous pouvons décider si r a lieu dans une séquence de règles tirable de $\mathcal{S}$.*

Démonstration. La règle r a lieu dans une séquence de règles tirable de $\mathcal{S}$ si, et seulement si, la transition correspondante t dans $\mathcal{N}$ apparaît dans une séquence de règles tirable de $\mathcal{N}$. Ceci est équivalent au fait de vérifier que le marquage de ${}^\bullet t$ est *couvert* par un marquage atteignable par $\mathcal{N}$. □

Première partie

Vérification de MSG à l'aide d'un solveur SAT

Chapitre

3 Introduction

Les MSC (pour « Messages Sequence Charts ») sont un modèle souvent utilisé pour la documentation des protocoles de télécommunication et qui a fait l'objet de nombreux travaux ces dernières années (citons par exemple les thèses [22], [49], [58] et [92]). Ils bénéficient d'une représentation visuelle et textuelle normalisée (ITU-T recommandation Z.120) et sont proches d'autres formalismes tels que les diagrammes de séquence de UML. Un MSC donne une description graphique des communications entre des processus. Habituellement, ce modèle fait abstraction des contenus des messages [11, 18, 19, 23, 50, 51, 75, 79, 84].

Les MSG (pour « Message Sequence Graphs »), ou de manière équivalente les HMSC (pour « High-level MSC »), sont un formalisme utilisé pour décrire des ensembles éventuellement infinis de scénarios. Ils se représentent simplement sous la forme d'un graphe orienté avec des nœuds étiquetés par des MSC. D'après le théorème de Kleene, ils sont équivalents à des expressions rationnelles dans le monoïde des MSC et correspondent ainsi à des ensembles rationnels de MSC. De nombreux outils ont déjà été développés afin de spécifier les protocoles sous forme de MSG. Certains d'entre eux mettent en œuvre également des techniques de vérification formelle, voir par exemple [19, 23, 6, 60, 21]. Dans cette première partie, nous nous intéressons principalement à la détection de la *divergence* de processus, introduite dans [18] et à la vérification de la *coopération globale* [50, 51] pour un MSG donné. Nous considérons également des propriétés d'accessibilité classiques, comme le caractère borné et la couverture, sur le contenu des canaux au cours des exécutions. En particulier, nous nous pencherons sur le problème du calcul de la *taille de la mémoire tampon* optimale des canaux, comme étudié dans [75]. Être en mesure de vérifier instantanément les propriétés de base des MSG que nous considérons pour des MSG relativement petits est intéressant. Cependant, comme les MSG peuvent être définis de façon modulaire, la vérification de grands graphes de manière efficace est également intéressante.

Comme les systèmes distribués asynchrones ne fournissent aucune information sur la vitesse relative des processus ou le temps de transit d'un message, de la *divergence* peut apparaître dans les spécifications : cela signifie qu'il n'y a pas de borne au nombre de messages en transit dans les canaux de communication. Toutefois, un critère simple nous permet de décider si un MSG donné n'est pas divergent [18, Th. 5] : nous devons vérifier que toutes les composantes connexes du graphe de communication de chaque cycle sont fortement connexes. Dans [18], Ben-Abdallah et Leue dérivent de leur critère un algorithme pour vérifier la divergence en temps quadratique dans le nombre de processus, mais exponentiel dans le nombre d'états. Il faut simplement rechercher tous les cycles *élémentaires* et calculer les composantes fortement connexes des graphes de communication [101]. Il s'ensuit que la détection de la divergence est dans NP. Cependant, un algorithme alternatif intéressant a été suggéré dans [11]. Cet algorithme consiste à fixer d'abord un canal potentiellement divergent et une partition des processus puis à rechercher un cycle élémentaire dans les composantes fortement connexes. Cette seconde approche est quadratique dans le nombre

d'états, mais exponentielle dans le nombre de processus. Ces deux algorithmes simples sont en quelque sorte opposés l'un de l'autre, mais chacun peut être efficace dans certains cas, selon le nombre d'états et de processus. Pour cette raison, la conception d'un algorithme efficace pour le détection de la divergence en général peut être difficile.

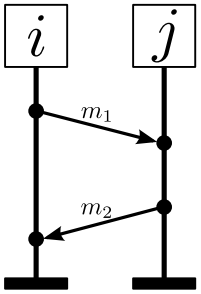
Détecter la divergence d'un MSG est connu pour être NP-complet [11, Th. 7]. Dans cette première partie, nous présentons une réduction quadratique de la divergence au problème de la satisfaisabilité booléenne (SAT) qui nous permet d'utiliser des SAT-solveurs pour détecter la divergence d'un MSG. De cette façon, *nous profitons de tous les travaux sur les SAT-solveurs* et pouvons détecter efficacement la divergence dans un MSG. Cette affirmation est confirmée par des expériences avec de petits et de grands MSG qui montrent que le *coût* de la réduction de SAT semble être *constant* en pratique.

Une autre propriété essentielle des MSG, appelée *coopération globale* [50, 51], a été étudiée dans la littérature : elle exige que le graphe de communication de chaque cycle soit connexe. De même que pour la non-divergence, la vérification de la coopération globale est co-NP-complet et deux algorithmes simples peuvent être proposés pour résoudre ce problème. Les MSG globalement coopératifs et non divergents sont souvent appelés localement synchronisés [84] ou bornés [11]. Ils bénéficient de techniques de model-checking spécifiques et sont connus pour être implémentables par des systèmes de transmission de messages [59]. En outre, les MSG globalement coopératifs correspondent également à des systèmes de transmission de messages pourvus de tampons non bornés [50]. C'est la raison pour laquelle la coopération globale est une propriété importante à vérifier lors de la conception d'un protocole avec des MSG. Par ailleurs, les MSG globalement coopératifs jouissent d'autres propriétés intéressantes. Par exemple, il est indécidable de savoir si deux MSG correspondent au même langage de MSC, mais ce problème devient décidable pour des MSG globalement coopératifs. Comme pour la divergence, nous présentons une réduction quadratique de la coopération globale vers SAT qui nous permet d'utiliser des SAT-solveurs pour vérifier la coopération globale.

Nous considérons également dans cette partie des problèmes d'accessibilité pour des MSG non divergents. Comme un MSC peut représenter un nombre exponentiel de scénarios, l'analyse du contenu d'un canal le long des exécutions potentielles s'avère difficile. Par exemple, une question naturelle est de calculer la taille des tampons de sorte que n'importe quel message en attente peut être stocké dans le système avant qu'il ne soit livré. Comme l'ont établi Lohrey et Muscholl, vérifier si la taille de la mémoire tampon est assez grande pour tous les scénarios d'un MSG (éventuellement divergent) est co-NP-complet [75, Th. 4.6]. Comme le MSG utilisé dans la démonstration de ce théorème n'a pas de cycle, ce résultat s'étend au cas particulier de MSG non divergent (ou globalement coopératif). Ainsi, le calcul d'une taille de tampon optimale pour un MSG non divergent est également difficile. Dans cette partie, nous nous intéressons à vérifier si une répartition de messages dans les canaux est atteignable, ou tout simplement couverte, au cours d'une exécution d'un MSG donné. Nous observons que ces deux problèmes sont NP-complets. Nous montrons en particulier qu'il suffit d'explorer tous les chemins de longueur bornée par un polynôme en la taille du MSG. Cela nous permet de réduire à SAT les propriétés de l'accessibilité et de la couverture.

En particulier, cette technique peut être utilisée pour calculer la taille minimal de la mémoire tampon pour un canal donné.

Cette partie est organisée de la manière suivante. Dans le reste de ce chapitre, nous commençons par définir formellement les MSG que nous considérons. Ensuite, dans le chapitre 4, nous introduisons les trois propriétés que nous voulons vérifier ainsi que la notion de graphe de communication. Le chapitre 5 présente les réductions à SAT que nous utilisons pour vérifier la non-divergence, la coopération globale et les problèmes d'accessibilité. Au cours des travaux de cette thèse, nous avons mis au point et comparés en pratique plusieurs approches. Nous présentons dans ce chapitre les meilleures formulations que nous avons obtenues. Pour terminer, dans le chapitre 6, nous évaluons expérimentalement nos algorithmes et les comparons à des outils existants.



(a)

```
msc Exemple;
  instance i;
    out m1 to j;
    in m2 from j;
  endinstance;
  instance j;
    in m1 from i;
    out m2 to i;
  endinstance;
endmsc;
```

(b)

FIGURE 3.1 – Exemple d'un MSC.

3.1 Les MSC

Les MSC (pour « Message Sequence Charts ») sont un formalisme bien connu pour décrire les scénarios d'interactions entre sites distants. Tout comme [24, 50, 59] nous adoptons trois restrictions pour les MSC : nous interdisons les actions internes ; le contenu des messages est abstrait ; tous les messages sont reçus dans un ordre First-In-First-Out (FIFO). De plus, les contraintes temporelles sont proscrites. Suivant [50, 23], nous adoptons l'idée des *MSC compositionnels* introduits dans [56] comme un moyen d'augmenter la puissance expressive du cadre formel et le domaine d'application de nos résultats. Ainsi, nous n'exigeons pas qu'à chaque émission dans un MSC corresponde une réception dans ce MSC. Afin de simplifier la sémantique des MSG compositionnels et de clarifier les preuves de certains résultats, nous reprenons l'approche proposée dans [15] et munissons les MSC d'informations additionnelles qui décrivent le contenu initial des canaux. De cette façon, nous bénéficions d'une notion formelle de la concaténation des MSC et un cadre algébrique clair.

Les MSC sont définis formellement par la recommandation Z.120 de l'ITU-T [63] avec une syntaxe formelle et des règles graphiques (figure 3.1). Ils peuvent être considérés également comme des mots partiels particuliers sur un alphabet que nous présentons en premier. Soit $\mathcal{I}$ un ensemble fini d'*instances* (aussi appelés « processus »). Pour toute instance $i \in \mathcal{I}$, l'alphabet Σ_i est l'union disjointe de l'ensemble des *actions d'émission* $\Sigma_i^! = \{i!j \mid j \in \mathcal{I} \backslash \{i\}\}$

et de l'ensemble des *actions de réception* $\Sigma_i^? = \{i?j \mid j \in \mathcal{I} \setminus \{i\}\}$. L'action $i!j$ désigne une émission de i vers j et $j?i$ la réception correspondante. Les alphabets Σ_i sont disjoints et nous posons $\Sigma_{\mathcal{I}} = \bigcup_{i \in \mathcal{I}} \Sigma_i$. Étant fixée une action $a \in \Sigma_{\mathcal{I}}$, nous noterons $\mathrm{Ins}(a)$ l'instance unique i telle que $a \in \Sigma_i$, qui est l'instance sur laquelle chaque occurrence de l'action a apparaît. Enfin, pour chaque mot partiel $(E, \preceq, \xi)$ sur $\Sigma_{\mathcal{I}}$ et pour tout $e \in E$, nous noterons $\mathrm{Ins}(e)$ l'instance sur laquelle l'événement e apparaît : $\mathrm{Ins}(e) = \mathrm{Ins}(\xi(e))$.

Nous notons $\mathcal{K} = \{(i,j) \in \mathcal{I} \times \mathcal{I} \mid i \neq j\}$ l'ensemble de tous les canaux entre les instances de $\mathcal{I}$. Un *marquage* indique le nombre de messages en transit à un moment de l'exécution : c'est formellement une application $\chi : \mathcal{K} \to \mathbb{N}$. Le marquage vide $\bar{0}$ met en correspondance chaque canal vers 0. Soit χ un marquage et $M = (E, \preceq, \xi)$ un mot partiel sur $\Sigma_{\mathcal{I}}$. Nous disons que deux événements $e, f \in E$ sont liés par rapport au marquage initial χ si e est un événement d'émission de i vers j et f la réception de ce message sur j selon l'hypothèse de communication FIFO. Formellement, nous notons $e \rightsquigarrow_\chi f$ s'il y a deux instances $i, j \in \mathcal{I}$ de telle sorte que $\xi(e) = i!j$, $\xi(f) = j?i$, et que $\chi(i,j) + \#^{i!j}(\downarrow_M e) = \#^{j?i}(\downarrow_M f)$. Cela signifie qu'il y a initialement $\chi(i,j)$ messages en attente allant de i vers j et e est la k-ième émission supplémentaire de i vers j, avec $k = \#^{i!j}(\downarrow_M e)$, et f est la l-ième réception sur le canal de i vers j avec $l = k + \chi(i,j)$.

Étant donné que chaque composant du mot partiel M peut être représenté par une structure de données de taille $O(|E|)$, il convient de définir la taille de $|M|$ par $|E|$ comme dans [75].

Définition 3.1.1. *Un MSC (pour « Message Sequence Chart ») est un couple (M, χ) où $M = (E, \preceq, \xi)$ est un mot partiel sur $\Sigma_{\mathcal{I}}$ et $\chi \in \mathbb{N}^{\mathcal{K}}$ est un marquage tel que*

M1 $\forall e, f \in E : \mathrm{Ins}(e) = \mathrm{Ins}(f) \Rightarrow (e \preceq f \vee f \preceq e)$

M2 $\forall e, f \in E : e \rightsquigarrow_\chi f \Rightarrow e \preceq f$

M3 $\forall e, f \in E : [e \prec\!\!\!\cdot\, f \wedge \mathrm{Ins}(e) \neq \mathrm{Ins}(f)] \Rightarrow e \rightsquigarrow_\chi f$

M4 $\forall (i,j) \in \mathcal{K} : \chi(i,j) + \#^{i!j}(M) \geq \#^{j?i}(M)$

Un MSC (M, χ) est également noté $M@\chi$. Par **M1**, les événements qui se produisent sur une même instance sont ordonnés de manière linéaire : des choix non déterministes ne peuvent pas être décrits dans un MSC. L'axiome **M2** formalise le fait que la réception d'un message n'apparaît qu'après l'événement d'émission correspondant. Par **M3**, la causalité dans M correspond à la dépendance linéaire sur chaque instance et l'ordonnancement des couples émission/réception. L'axiome **M4** signifie intuitivement qu'à chaque événement de réception dans M correspond soit un message en attente dans χ soit un événement d'émission dans M.

Définition 3.1.2. *Soit $M@\chi$ un MSC. Alors le marquage χ est appelé la* source *de $M@\chi$. La* cible *de $M@\chi$ est le marquage χ' tel que $\chi'(i,j) = \chi(i,j) + \#^{i!j}(M) - \#^{j?i}(M)$ pour tous les canaux $(i,j) \in \mathcal{K}$. Un MSC est dit* basique *si la source et la cible correspondent au marquage vide $\bar{0}$.*

Notons que l'axiome **M4** de la définition 3.1.1 assure que la cible d'un MSC est un marquage.

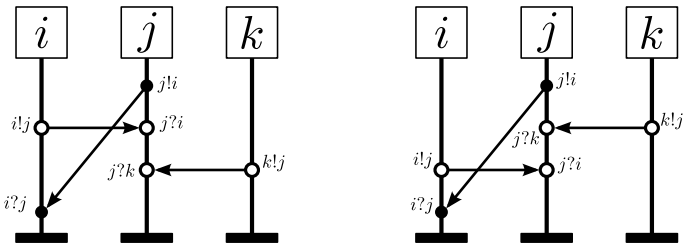


FIGURE 3.2 – Deux descriptions graphiques d'un même MSC.

Les MSC basiques correspondent à la notion habituelle des MSC. Ils admettent une description graphique normalisée où chaque instance est représentée par une ligne verticale et des événements d'émission ou de réception. Chaque communication est représentée par une flèche entre deux instances (figure 3.1). Les MSC plus généraux que nous considérons ici peuvent être dessinés de manière similaire aux MSC basiques en ajoutant à l'ensemble des événements réels, des événements *fictifs*. Par exemple la figure 3.2 montre deux MSC avec deux événements réels représentés avec des cercles pleins et quatre événements fictifs qui sont représentés par des cercles vides. Le rôle des émissions fictives est de spécifier la source associée au MSC. Nous exigeons que les émissions fictives forment un *préfixe*, ce qui signifie que tout événement qui apparaît avant une émission fictive est une émission fictive (sur cette même instance). De la même manière, les réceptions fictives forment un suffixe : tout événement qui apparaît après une réception fictive sur une même instance est une réception fictive. La figure 3.2 montre qu'il peut y avoir plusieurs descriptions graphiques non isomorphes d'un même MSC en fonction de l'ordre relatif des événements fictifs.

3.2 Produit des MSC

Les MSC basiques bénéficient d'un produit naturel illustré dans la figure 3.3. Soit deux MSC basiques M_1 et M_2, alors le MSC basique $M_1 \cdot M_2$ consiste à coller les deux ordres partiels l'un après l'autre le long des lignes d'instance. Nous montrons ici que les MSC généraux peuvent également être équipés d'un produit qui étend le produit habituel des MSC basiques.

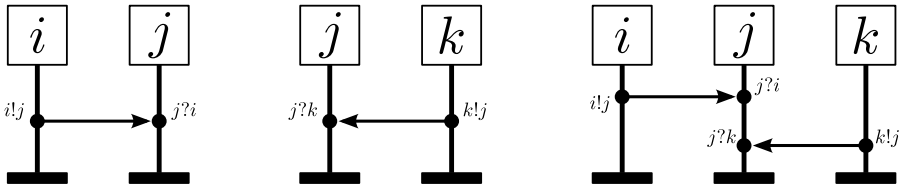


FIGURE 3.3 – Produit de deux MSC basiques.

Nous observons tout d'abord que les MSC satisfont la *propriétés de consistance* suivante : si deux MSC partagent la même source et une même extension linéaire alors elles sont identiques.

Propriété 3.2.1. *Soit $M@\chi$ et $M'@\chi'$ deux MSC. Si $\chi = \chi'$ et $\mathrm{LE}(M) \cap \mathrm{LE}(M') \neq \emptyset$ alors $M = M'$.*

Démonstration. Soit $s \in \mathrm{LE}(M) \cap \mathrm{LE}(M')$ une extension linéaire commune de M et M'. L'ordre linéaire sur chaque instance dans M peut être récupéré à partir de s. En outre, l'ordre partiel des événements de M est alors caractérisé par l'axiome **M3** et donc peut être extrait de χ. Cela reste vrai pour M'. Ainsi, $M = M'$. □

En d'autres termes, nous pouvons récupérer un MSC à partir de sa source et de l'une de ses extensions linéaires.

L'affirmation suivante formalise comment nous pouvons concaténer deux MSC l'un après l'autre à condition que la cible du premier coïncide avec la source du second. Le MSC résultant partage sa source avec le premier MSC et sa cible avec le deuxième. En outre, toute exécution linéaire constituée d'une extension linéaire du premier MSC suivie d'une extension linéaire du second MSC est une extension linéaire du MSC résultant.

Propriété 3.2.2. *Soit $M_1@\chi_1 = (E_1, \preceq_1, \xi_1, \chi_1)$ et $M_2@\chi_2 = (E_2, \preceq_2, \xi_2, \chi_2)$ deux MSC tels que la cible de $M_1@\chi_1$ est χ_2. Soit $\rightsquigarrow$ la relation binaire sur $E_1 \times E_2$ tel que $e_1 \rightsquigarrow e_2$ si $\xi_1(e_1) = i!j$, $\xi_2(e_2) = j?i$, et $\chi_1(i,j) + \#^{i!j}(\downarrow_{M_1} e_1) = \#^{j?i}(M_1) + \#^{j?i}(\downarrow_{M_2} e_2)$. Alors le quadruplet $(E, \preceq, \xi, \chi_1)$ où $E = E_1 \uplus E_2$, $\xi = \xi_1 \cup \xi_2$ et l'ordre partiel $\preceq$ est la fermeture transitive de $\preceq_1 \cup \preceq_2 \cup \{(e_1, e_2) \in E_1 \times E_2 \mid \mathrm{Ins}(e_1) = \mathrm{Ins}(e_2)\} \cup \rightsquigarrow$ est un MSC $M@\chi_1$. Par ailleurs $\mathrm{LE}(M_1).\mathrm{LE}(M_2) \subseteq \mathrm{LE}(M)$.*

Démonstration. Il est facile de vérifier que les quatre exigences de la définition 3.1.1 sont satisfaites par $M@\chi$. Ce dernier est donc un MSC. De plus, aucun événement de E_2 est au-dessous d'un événement de E_1. Il s'ensuit que $\mathrm{LE}(M_1).\mathrm{LE}(M_2) \subseteq \mathrm{LE}(M)$. □

Considérons deux extensions linéaires $u_1 \in \mathrm{LE}(M_1)$ et $u_2 \in \mathrm{LE}(M_2)$. Alors le mot concaténé $u_1.u_2$ est une extension linéaire de M. Ainsi, toute observation séquentielle composée d'une vision linéaire de $M_1@\chi_1$ suivie d'une vision linéaire de $M_2@\chi_2$ est une vision linéaire potentielle $M@\chi_1$. Cela est illustré dans la figure 3.3 dans le cas simple de deux MSC basiques. Notons que les événements de M_2 peuvent se produire avant les événements de M_1 dans une extension linéaire de $M_1 \cdot M_2$. Par exemple, l'événement étiqueté par $k!j$ peut se produire avant l'événement étiqueté $i!j$ bien que $k!j$ apparaît dans M_2 et $i!j$ apparaît dans M_1. Pour cette raison, la concaténation de MSC est souvent appelée *produit asynchrone*.

Nous arrivons maintenant à la définition générale de la concaténation de deux MSC. Nous commençons par ajouter un MSC spécial désigné par 0 à la classe des MSC. Ce MSC supplémentaire 0 est dit *non-valide* et agira comme un zéro : nous posons $x \cdot 0 = 0 \cdot x = 0$ pour tout MSC x.

Définition 3.2.3. *Soit $M_1@\chi_1 = (E_1, \preceq_1, \xi_1, \chi_1)$ et $M_2@\chi_2 = (E_2, \preceq_2, \xi_2, \chi_2)$ deux MSC valides. Soit u_1 et u_2 des extensions linéaires de M_1 et M_2 respectivement. Si la cible de $M_1@\chi_1$ est χ_2 alors le produit $M_1@\chi_1 \cdot M_2@\chi_2$ est le MSC valide $M@\chi_1$ tel que $u_1.u_2 \in \mathrm{LE}(M)$. Sinon, nous posons $M_1@\chi_1 \cdot M_2@\chi_2 = 0$.*

Supposons que la cible de $M_1@\chi_1$ est égale à la source de $M_1@\chi_2$. Alors la propriété 3.2.2 garantit l'existence d'un MSC $M@\chi_1$ tel que $u_1.u_2 \in \mathrm{LE}(M)$. Par ailleurs, la propriété 3.2.1 garantit qu'un tel MSC est unique. Ainsi, le produit de deux MSC est bien défini. Notez également que la spécification de l'ordre partiel des événements dans ce produit de MSC se caractérise par la propriété 3.2.2. Il est facile de vérifier à partir de la définition 3.2.3 et la propriété 3.2.1 que le produit d'MSC est associatif. Par conséquent, le produit de n'importe quelle séquence de MSC est bien défini (mais peut être égal au MSC non valide).

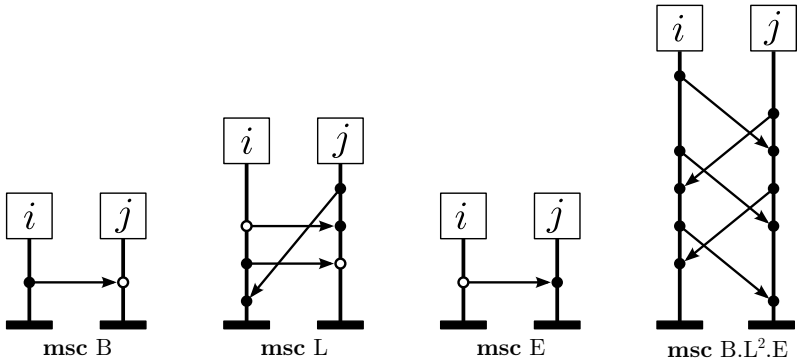


FIGURE 3.4 – Protocole du bit alternant $B \cdot L^\star \cdot E$.

Exemple 3.2.4. *Considérons les MSC B, L, et E sur le côté gauche de la figure 3.4. Alors $B \cdot E$ est un MSC basique alors que le produit $E \cdot B$ est égal au MSC non valide ($E \cdot B = 0$). Le MSC basique $B \cdot L \cdot L \cdot E$ est représenté sur la partie droite de la figure 3.4.*

Soit $\mathbb{MSC}$ la classe de tous (valides et non valides) les MSC. La classe de MSC forme un semi-groupe. Nous pouvons identifier formellement tout MSC vide comme un seul MSC et obtenir un monoïde.

Remarque 3.2.5. *Soit $M_1@\bar{0} = (E_1, \preceq_1, \xi_1, \bar{0})$ et $M_2@\bar{0} = (E_2, \preceq_2, \xi_2, \bar{0})$ deux MSC basiques. Alors le produit $M_1@\bar{0} \cdot M_2@\bar{0}$ est le MSC basique $(E, \preceq, \xi, \bar{0})$ où $E = E_1 \uplus E_2$, $\xi = \xi_1 \cup \xi_2$ et l'ordre partiel $\preceq$ est la fermeture transitive de $\preceq_1 \cup \preceq_2 \cup \{(e_1, e_2) \in E_1 \times E_2 \mid \mathrm{Ins}(e_1) = \mathrm{Ins}(e_2)\}$. Cet ordre coïncide avec la concaténation habituelle des MSC basiques. Ainsi, le produit de la définition 3.2.3 étend le produit asynchrone des MSC basiques.*

3.3 Les MSG

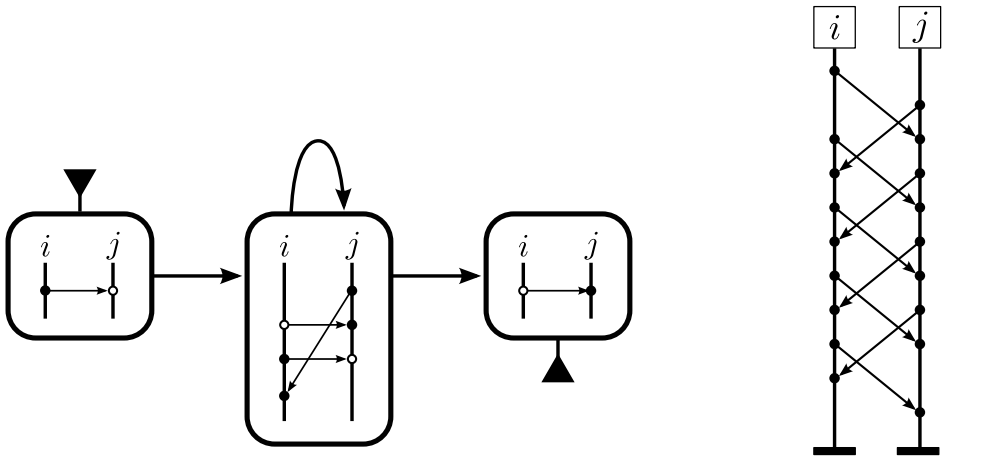
Les MSC de haut niveau (ou HMSC pour « High-level MSC ») sont des expressions rationnelles construites à partir d'ensembles finis de MSC en utilisant les opérations d'union, de produit et d'itération.

Exemple 3.3.1. *Considérons à nouveau les MSC B, L et E de la figure 3.4. Un MSC basique typique associé au HMSC $B \cdot L^\star \cdot E$ est représenté sur la partie droite de la figure 3.4.*

Nous nous intéressons ici aux parties rationnelles de MSC valides. Par le théorème de Kleene, ces langages peuvent être décrits par un graphe orienté dont les nœuds sont étiquetés par des MSC. Ces structures sont souvent appelées des MSG [11, 23, 59], des « MSC specifications » [18], des HMSC [56, 82, 75, 51, 23], ou des « MSC-graphs » [84, 10, 50].

Définition 3.3.2. *Un* MSG *(pour « Message Sequence Graph ») est un quadruplet* $\mathcal{G} = (Q, \imath, A, \mu)$ *qui se compose d'un ensemble fini d'états* Q *avec un état initial* $\imath \in Q$*, un ensemble d'arcs* $A \subseteq Q \times Q$*, et un étiquetage* $\mu : Q \to \mathbb{MSC}$ *qui associe à chaque état un MSC* valide*. Il est nécessaire que pour chaque arc* a *allant d'un état* q_1 *à un état* q_2 la source du MSC $\mu(q_2)$ soit égale à la cible du MSC $\mu(q_1)$.

Notons que tout chemin $q_0 \to ... \to q_n$ correspond à un MSC valide $\mu(q_0) \cdot ... \cdot \mu(q_n)$. Un exemple d'un MSG décrivant le protocole du bit alternant est illustré par la figure 3.5.



(a) Modélisation avec un MSG (b) Un MSC accepté

FIGURE 3.5 – Protocole du bit alternant sous la forme d'un MSG.

Définition 3.3.3. *Le langage de MSC* $\mathcal{L}(\mathcal{G})$ *reconnu par le* MSG $\mathcal{G}$ *recueille tous les MSC* $M@\chi$ *pour lesquels il existe un chemin* $\imath = q_0 \to ... \to q_n$ *tel que* $M@\chi = \mu(q_0) \cdot ... \cdot \mu(q_n)$.

Pour plus de commodité, nous supposons que tous les états d'un MSG sont accessibles à partir de son état initial. Les MSG sont généralement pourvus d'un sous-ensemble $F \subseteq Q$ d'états finaux. Dans ce cas, la définition de $\mathcal{L}(\mathcal{G})$ exige en plus que $q_n \in F$ et nous aurions besoin alors que chaque état soit co-accessible depuis les états finaux. Cependant comme cette fonction ne joue aucun rôle dans notre étude, nous ne la prenons pas en compte. Notons que la taille d'un MSG peut être définie par la somme des tailles :

— du graphe : soit $|Q|^2$ en utilisant une représentation par matrice,

— des étiquettes : soit $\sum_{q \in Q}(log_2(|Q|) + |\mu(q)|) = |Q| \times log_2(|Q|) + \sum_{q \in Q} |\mu(q)|$,

— de l'état initial : soit $log_2(|Q|)$.

Ainsi, la taille asymptotique d'un MSG peut être définie par la somme des tailles des MSC de chaque état plus la taille du graphe.

3.4 Lien avec les MSC compositionnels

Les MSC compositionnels (aussi appelés « compositional MSC » ou cMSC) introduits dans [56] correspondent aux MSC de la définition 3.1.1 mais dans lesquels la source est omise. Plusieurs sources peuvent correspondre à un cMSC donné de sorte que la concaténation d'une séquence de cMSC conduit à un *ensemble* de cMSC potentiels [50, Def. 2.5]. Cet ensemble peut être vide et correspond alors intuitivement au MSC non valide que nous avons introduit. Un cMSC-graphe est un graphe orienté dont les nœuds sont étiquetés par un cMSC. Ils sont pourvus d'un sous-ensemble de nœuds de départ et un sous-ensemble de nœuds finaux. Sans perte de généralité, on peut supposer qu'il n'y a qu'un seul nœud initial et qu'un seul nœud final et qu'ils sont tous les deux étiquetés par le cMSC vide. Dans [56], il est exigé qu'il n'y ait pas de message en attente au début d'un MSC accepté. En d'autres termes, le marquage source implicite de tous les nœuds initiaux est le marquage vide. Un cMSC-graphe est dit *sûr* [50, Def. 3.1] si pour tout chemin allant d'un nœud initial vers un nœud final, la concaténation des cMSC sur ce chemin contient un MSC *basique*, c'est-à-dire qu'il n'y a aucun message non reçu. L'ensemble des MSC basiques obtenus de cette façon est appelé le langage reconnu par le cMSC-graphe. Cela signifie que la cible implicite des nœuds finaux est le marquage vide. En outre deux chemins partant d'un nœud initial vers un même nœud donné, doivent aboutir au même marquage cible (dès lors que chaque nœud est co-accessible à partir d'un nœud final). Par conséquent, chaque cMSC-graphe sûr peut être considéré formellement comme un MSG tel que défini dans la définition 3.3.2. En d'autres termes, la notion de MSG que nous considérons dans cette thèse est équivalente à la notion de cMSC-graphe sûr (hormis le fait anecdotique que nous n'imposons pas que le marquage des nœuds initiaux ou finaux soit vide).

Chapitre

4 Propriétés de base à vérifier

Les MSG servant généralement à modéliser des protocoles de communication, la vérification de la non-divergence [18] et de la coopération globale [50, 51] permet d'écarter certaines difficultés lors de l'implémentation ou de l'exécution du protocole. Connaître la taille des tampons peut également être utile afin de dimensionner correctement les canaux de communication [75]. Ces trois propriétés ont été souvent étudiées dans la littérature. Nous rappelons leur définition dans ce chapitre. Afin de vérifier que certaines situations n'apparaissent pas ou qu'au contraire une configuration est possible, nous introduisons également les problématiques de la couverture d'un marquage et de l'accessibilité.

4.1 Accessibilité et couverture par préfixe

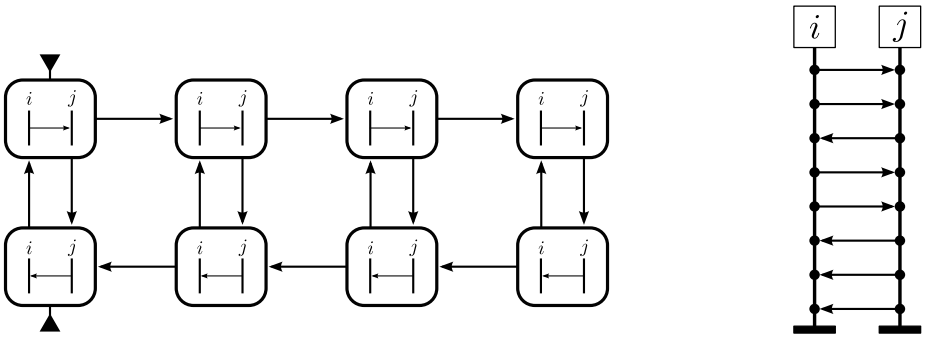
Soit $M@\chi = (E, \preceq, \xi, \chi)$ un MSC. Une exécution de $M@\chi$ est une vue séquentielle des événements, à savoir, une extension linéaire $s = (E, \leq, \xi)$ de M. Chaque étape de cette exécution correspond à un mot préfixe $s' \leq s$. Le marquage χ' atteint après s' est tel que $\chi'(i,j) = \chi(i,j) + \#^{i!j}(s') - \#^{j?i}(s')$ pour chaque canal $(i,j) \in \mathcal{K}$. Cela signifie que le nombre de messages dans le canal (i,j) après l'exécution s' est égal au nombre de messages initialement dans ce canal ou envoyés dans ce canal le long de s' et qui n'ont pas été reçus le long de s'.

Définition 4.1.1. *Un marquage χ_0 est* accessible par préfixe *(ou* préfixe-accessible*) dans un MSG $\mathcal{G}$ s'il existe un MSC $M@\chi \in \mathcal{L}(\mathcal{G})$, une extension linéaire $s \in \mathrm{LE}(M)$ et un mot préfixe $s' \leq s$ tel que $\chi_0(i,j) = \chi(i,j) + \#^{i!j}(s') - \#^{j?i}(s')$ pour chaque canal $(i,j) \in \mathcal{K}$.*

Dans ce cas, nous pouvons considérer le mot partiel M' constitué du sous-ensemble d'événements de s' et obtenir un MSC $M'@\chi$ dont la cible est χ_0. Il est clair qu'un marquage est accessible par préfixe dans $\mathcal{G}$ si, et seulement si, il est la cible d'un préfixe d'un MSC accepté.

La première propriété naturelle que nous aimerions vérifier consiste à décider si un marquage donné est accessible par préfixe dans un MSG donné. Nous étudierons aussi le problème voisin suivant : le problème de *couverture par préfixe*, pour un MSG $\mathcal{G}$ donné et un marquage χ donné, consiste à savoir s'il existe un marquage préfixe-accessible χ' tel que $\chi(i,j) \leq \chi'(i,j)$ pour tous les canaux $(i,j) \in \mathcal{K}$. Cela correspond au problème de couverture habituel pour les réseaux de Petri.

Décider si un marquage donné est accessible par préfixe correspond à une question naturelle quand on essaie de comprendre un protocole. Le problème de couverture semble aussi être très utile dans la pratique. Par exemple, il nous permet de vérifier s'il peut y avoir *au moins* quatre messages dans un canal particulier (i,j). Cette situation correspond au fait que le marquage χ_0 tel que $\chi_0(i,j) = 4$ et $\chi_0(k,l) = 0$ pour tous les autres canaux est couvert par un marquage accessible par préfixe.



(a) Modélisation avec un MSG (b) Un MSC accepté

FIGURE 4.1 – Protocole des fenêtres coulissantes simplifié (fenêtre de taille 3).

Exemple 4.1.2. *Considérons le protocole des fenêtres coulissantes décrit par le MSG représenté dans la figure 4.1 pour lequel l'étiquette de chaque état est un MSC basique qui est constitué d'un seul échange de message. L'état initial est représenté graphiquement par un triangle noir dirigé vers le bas et l'état final est représenté par un triangle noir dirigé vers le haut. Résoudre le problème de couverture nous permet d'affirmer qu'il n'y a jamais plus de trois messages dans le canal* (i, j).

La notion d'accessibilité par préfixe porte sur le contenu des canaux mais néglige les états de contrôle. Toutefois, il est possible de vérifier cet aspect avec l'ajout d'instances observatrices. Considérons à nouveau le protocole des fenêtres coulissantes simplifié de la figure 4.1. Est-il possible que l'instance i soit dans l'état en bas à gauche, tandis que l'instance j est située dans l'état en haut à droite ? Pour répondre à cette question, nous introduisons deux nouvelles instances i' et j' et considérons les deux MSC I et J de la figure 4.2. Nous considérons le nouveau MSG $\mathcal{G}'$ obtenu en remplaçant le MSC de l'état en bas à gauche par le MSC I et le MSC en haut à droite par le MSC J. Alors la question revient à savoir si les canaux (i, i') et (j, j') peuvent contenir un message en même temps, c'est-à-dire si le marquage χ_0 tel que $\chi_0(i, i') = 1$, $\chi_0(j, j') = 1$ et $\chi_0(k, l) = 0$ pour tous les autres canaux est couvert par un marquage accessible par préfixe.

FIGURE 4.2 – MSC I à gauche et MSC J à droite.

Nous pouvons adapter la construction de [75, Th. 4.5] pour montrer que les problèmes consistant à vérifier si un marquage est accessible ou couvert sont tous les deux des problèmes NP-difficiles.

Propriété 4.1.3. *L'accessibilité par préfixe et la couverture par préfixe sont des problèmes NP-difficiles.*

Démonstration. Afin de prouver la NP-difficulté, nous réduisons 3-SAT à nos problèmes. Soit $\{x_1, ..., x_n\}$ un ensemble de variables propositionnelles et $\Psi = \{C_1, ..., C_m\}$ un ensemble de clauses, où chaque clause C_i se compose de variables positives et de variables négatives. Nous considérons le MSG $\mathcal{G} = (Q, i, A, \mu)$ avec $Q = \{\imath, q_1, \bar{q}_1, ..., q_n, \bar{q}_n\}$ et $A = \{(\imath, q_1), (\imath, \bar{q}_1)\} \cup \{(q_i, q_{i+1}), (q_i, \bar{q}_{i+1}), (\bar{q}_i, q_{i+1}), (\bar{q}_i, \bar{q}_{i+1}) | 1 \leq i < n\}$. Hormis l'état initial, chaque état correspond à une variable ou sa négation. L'ensemble des instances est $\mathcal{I} = \{c_i, c'_i | 1 \leq i \leq m\}$, c'est-à-dire qu'il y a deux instances pour chaque clause. Le MSC $\mu(\imath)$ est vide. Le MSC $\mu(q_i)$ contient un message allant de l'instance c_j à l'instance c'_j et un second message de l'instance c'_j à l'instance c_j si le littéral x_i apparaît dans C_j. De même, le MSC $\mu(\bar{q}_i)$ contient un message allant de l'instance c_j à l'instance c'_j et un second message de l'instance c'_j à l'instance c_j si le littéral $\bar{x}_i$ apparaît dans C_j. Cette construction est illustrée par la figure 4.3. Soit un marquage χ_0 tel que $\chi_0(c'_j, c_j) = 1$ pour $1 \leq j \leq m$ et égal à zéro pour les autres canaux. Alors Ψ est satisfiable si, et seulement si, il existe un MSC $M@\chi \in \mathcal{L}(\mathcal{G})$ contenant un échange de messages entre c_j et c'_j pour tout $1 \leq j \leq m$. Cela est vrai si, et seulement si, χ_0 est accessible (ou couvert) dans $\mathcal{G}$. □

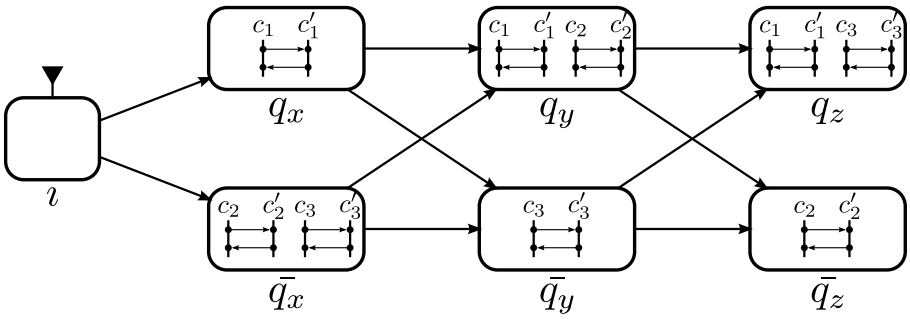


FIGURE 4.3 – Exemple de la réduction pour $(x \vee y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg x \vee \neg y \vee z)$.

Nous verrons dans la section 5.3 que ces deux problèmes sont NP-complets.

4.2 Canaux bornés et taille des mémoires tampons

Considérons à nouveau un MSC $M@\chi$ et une extension linéaire s de M. Le nombre maximal de messages en attente dans le canal allant de i vers j le long d'une extension linéaire s d'un MSC $M@\chi$ est $\max_{s' \leq s}(\chi(i,j) + \#^{i!j}(s') - \#^{j?i}(s'))$. Comme cette valeur dépend de s, la *largeur* de $M@\chi$ pour un canal (i, j) est définie comme le maximum de cette valeur pour toutes les extensions linéaires de s : ainsi,

$$W_{i,j}(M@\chi) = \max_{s \in \mathrm{LE}(M)} \max_{s' \leq s} (\chi(i,j) + \#^{i!j}(s') - \#^{j?i}(s'))$$

représente le nombre maximal de messages en attente, à toute étape de l'exécution de $M@\chi$. Pour un MSG $\mathcal{G}$, nous considérons la borne supérieure de la largeur de tous les MSC acceptés

par $\mathcal{G}$: $W_{i,j}(\mathcal{G}) = \sup_{M@\chi \in \mathcal{L}(\mathcal{G})} W_{i,j}(M@\chi)$. Notons ici que le langage d'un MSC $\mathcal{L}(\mathcal{G})$ est infini lorsque $\mathcal{G}$ contient des cycles. Ainsi, $W_{i,j}(\mathcal{G})$ peut-être également infini. La valeur $W_{i,j}(\mathcal{G})$ est utile, car elle représente la taille minimale de la mémoire tampon du canal allant de i vers j de sorte qu'aucun dépassement de capacité n'apparaît dans l'exécution de $\mathcal{G}$.

Définition 4.2.1. *Un canal* (i, j) *est* borné *dans un MSG* $\mathcal{G}$ *si* $W_{i,j}(\mathcal{G}) < \infty$. *Un MSG* $\mathcal{G}$ *est à canaux bornés si tous les canaux sont bornés dans* $\mathcal{G}$.

À notre connaissance, calculer $W_{i,j}(\mathcal{G})$ efficacement n'a pas encore été abordé dans la littérature. Cependant, il est prouvé dans [75, Th. 4.6] que le calcul de la taille de la mémoire tampon optimale $\max_{(i,j)\in\mathcal{K}} W_{i,j}(\mathcal{G})$ est NP-difficile. Plus précisément, vérifier si une taille de mémoire tampon est plus grande que la taille optimale est co-NP-complet.

4.3 Divergence et largeur de canaux bornés

Un bon critère pour détecter le caractère borné d'un canal d'un MSG a été établi dans [18, Th. 5] dans le cas particulier des MSG utilisant seulement des MSC basiques. Il s'appuie sur la notion suivante de graphe de communication.

Définition 4.3.1. *[18, 51, 59] Le* graphe de communication *d'un MSC* $M@\chi$ *est le graphe orienté dont les nœuds sont les instances qui émettent ou reçoivent un message dans* M *(appelées les* instances actives *de* M*) et tel qu'il y a un arc de* i *vers* j *chaque fois que* M *comprend soit une émission soit une réception d'un message de* i *vers* j.

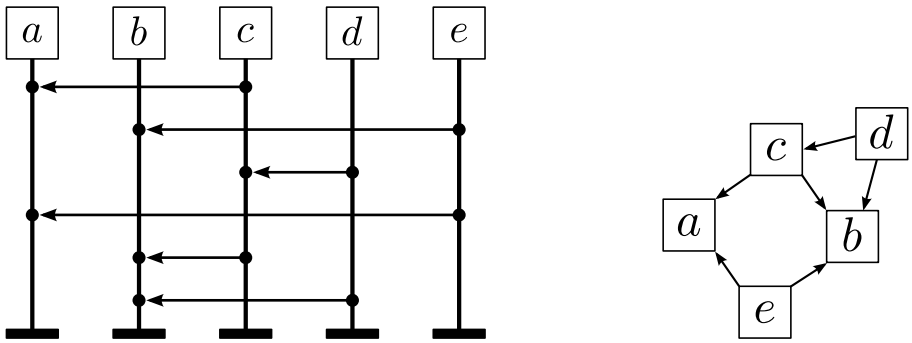
Par exemple, le graphe de communication du MSC de la figure 4.4(a) est représenté sur la figure 4.4(b). Remarquons ici que ce graphe est fortement connexe. Étant donné un MSG $\mathcal{G}$ et $Q' \subseteq Q$ un sous-ensemble d'états, le graphe de communication de Q' est tout simplement l'union des graphes de communication des MSC portés par les états de Q'. Le graphe de communication d'un sous-ensemble d'arcs $A' \subseteq A$ est le graphe de communication de l'ensemble des nœuds des domaines et des codomaines des arcs A'. En particulier, le graphe de communication d'un cycle de $\mathcal{G}$ est le graphe de communication du sous-ensemble d'états contenus dans ce cycle. Ce graphe de communication correspond également au graphe de communication de chaque MSC obtenu par le produit des MSC le long de ce cycle.

Pour plus de commodité, nous introduisons la notion de divergence pour un canal donné.

Définition 4.3.2. *Soit* $\mathcal{G}$ *un MSG. Un canal* (i, j) *est* divergent *s'il existe un cycle dont le graphe de communication contient un arc allant de* i *vers* j *mais pas de chemin de* j *vers* i. *Le MSG* $\mathcal{G}$ *est dit* divergent *si* au moins un *canal est divergent dans* $\mathcal{G}$.

Remarquons ici que le MSG est divergent si, et seulement si, il existe un cycle dont le graphe de communication contient une composante connexe qui n'est pas fortement connexe.

La propriété 5.1.1 montrera que l'on peut se restreindre aux cycles élémentaires. Par conséquent, la notion de divergence de la définition 4.3.2 coïncide avec le critère établi par Ben-Abdallah et Leue [18]. Ainsi, un MSG étiqueté par des MSC basiques est à canaux bornés si, et seulement si, il n'est pas divergent. Nous allons étendre cette caractérisation à tous les MSG dans le chapitre 5. Il est clair que le problème de la détection de la divergence est dans NP. En fait, ce problème est connu pour être NP-complet [11, Th. 7].



(a) (b)

FIGURE 4.4 – Un MSC et son graphe de communication.

4.4 Coopération globale et synthèse

Les MSG globalement coopératifs jouent un rôle important dans la théorie des MSC. D'une part, leurs langages peuvent être mis en œuvre sous la forme d'un système à transmission de messages avec des tailles de mémoires tampons possiblement non bornés, au moyen d'informations de contrôle ajoutées aux messages [51]. Par ailleurs si le MSG est également à canaux bornés alors ce système ne nécessite que des mémoires tampons bornées [59]. D'autre part, leurs langages sont définissables dans la logique MSO pour laquelle les propriétés d'inclusion sont décidables même si le langage n'est pas à canaux bornés [50]. De la même manière que la divergence, la notion de globalement coopératif consiste en une contrainte sur les graphes de communication des cycles.

Définition 4.4.1. *Un canal (i, j) est* globalement coopératif *dans un MSG $\mathcal{G}$ si i et j appartiennent à la même composante connexes dans le graphe de communication de tout cycle pour lequel à la fois i et j sont actifs. Un MSG $\mathcal{G}$ est dit* globalement coopératif *si* tous *les canaux sont globalement coopératifs dans $\mathcal{G}$.*

Notons que le canal (i, j) est globalement coopératif dès que (j, i) est globalement coopératif. Contrairement à la divergence qui est souvent considérée comme un inconvénient ou même une erreur dans la spécification, la coopération globale est une exigence habituelle afin de garantir un bon comportement du système. Pour cette raison, nous exigeons que tous les canaux soient globalement coopératifs.

Remarque 4.4.2. *Nous allons démontrer que les contraintes sur les graphes de communication pour le problème de la divergence peuvent être limitées aux cycles* élémentaires *(propriété 5.1.1). La situation est différente pour la coopération globale. Le fait de ne considérer que les cycles élémentaires conduit à une plus grande classe de MSG qui correspond à des langages de MSC qui ne sont pas nécessairement définissables en logique MSO, contrairement aux MSG globalement coopératifs [50]. Cette observation est essentiellement inspirée par [84, Exemple 1]. Nous considérons dans la figure 4.5 le MSG avec 5 états étiquetés par les MSC basiques A, B, et C. Ce MSG est évidemment non-divergent, mais il n'est pas globalement coopératif. Il est facile de voir que $\mathcal{L}(\mathcal{G})$ n'est pas définissable en MSO*

avec l'aide de la théorie des langages de MSC réguliers [59]. Nous procédons par contradiction. Supposons que $\mathcal{L}(\mathcal{G})$ *est définissable en MSO. Alors comme* $\mathcal{L}(\mathcal{G})$ *est à canaux bornés,* $\mathcal{L}(\mathcal{G})$ *est régulier [59, Th. 4.4]. Par conséquent, l'ensemble des extensions linéaires* $\mathrm{LE}(\mathcal{L}(\mathcal{G})) = \bigcup_{M@\bar{0}\in\mathcal{L}(\mathcal{G})} \mathrm{LE}(M)$ *est un ensemble de mots réguliers [59, Def. 2.2]. Alors, le langage de mots* $L' = \mathrm{LE}(\mathcal{L}(\mathcal{G})) \cap (\mathrm{LE}(A^2)^\star.\mathrm{LE}(B \cdot C)^\star)$ *est aussi régulier puisque* $\mathrm{LE}(A^2)$ *et* $\mathrm{LE}(B \cdot C)$ *sont finis, et par conséquent, ils sont réguliers. Toutefois, il est facile de se convaincre que* $L' = \{\mathrm{LE}(A^2)^n.\mathrm{LE}(B \cdot C)^n \mid n \geq 0\}$ *et qu'il n'est pas régulier.*

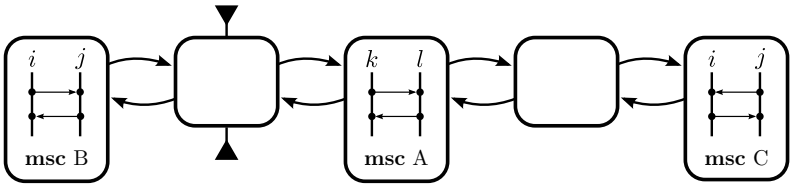


FIGURE 4.5 – MSG pour la remarque 4.4.2.

Chapitre

5 Utilisation de SAT-solveurs

Nous avons vu dans le chapitre précédent que la plupart des propriétés intéressantes à vérifier sur les MSG sont des problèmes NP-complets. Afin de pouvoir résoudre ces problèmes de manière efficace, nous allons les traduire en formule booléenne. En effet, de nombreuses recherches on été effectuées ces dernières années sur les SAT-solveurs si bien que la résolution de formule booléenne se trouve être étonnamment rapide en pratique.

Nous commençons par étudier le problème de la divergence et donnons une formulation booléenne simple. Nous étudions ensuite la coopération globale et donnons une formulation un peu plus complexe. Nous terminons ensuite par les problèmes liés à l'accessibilité par préfixe.

5.1 Divergence avec SAT

Nous fixons un MSG $\mathcal{G} = (Q, \imath, A, \mu)$. Afin de clarifier la présentation et de renforcer les similitudes et les différences entre la divergence et la coopération globale, les deux définitions prennent en compte des propriétés du graphe de communication de n'importe quel cycle dans $\mathcal{G}$. En ce qui concerne la divergence, la propriété suivante affirme que nous pouvons nous concentrer sur des *cycles élémentaires*. Ceci est très utile afin de vérifier la non-divergence d'un MSG donné. L'affirmation suivante stipule en outre que nous pouvons considérer un ensemble de cycles élémentaires disjoints, ce qui nous aidera à concevoir une formule booléenne simple pour le codage de la divergence.

Propriété 5.1.1. *Soit i et j deux instances. Les quatre propriétés suivantes sont équivalentes :*

1. *le canal (i, j) est divergent dans $\mathcal{G}$;*
2. *il existe un cycle dans $\mathcal{G}$ dont le graphe de communication contient un arc allant de i vers j mais pas de chemin allant de j vers i ;*
3. *il existe un cycle* élémentaire *dans $\mathcal{G}$ dont le graphe de communication contient un arc allant de i vers j mais pas de chemin allant de j vers i ;*
4. *il existe un ensemble d'arcs qui forment des* cycles élémentaires disjoints *et dont le graphe de communication contient un arc allant de i vers j mais pas de chemin allant de j vers i.*

Démonstration. Par définition (1) est équivalent à (2). Supposons que (2) soit vraie : il existe un cycle $q_0 \longrightarrow q_1 ... q_{n-1} \longrightarrow q_n = q_0$ dont le graphe de communication contient un chemin allant de i vers j mais pas de j vers i. Soit q un état de ce cycle tel que $\mu(q)$ contient un message allant de i vers j. Considérons maintenant un cycle élémentaire $q_k \longrightarrow q_{k+1} ... q_{l-1} \longrightarrow q_l = q_k$ qui contient q. Son graphe de communication contient un arc allant de i vers j mais pas de chemin allant de j vers i. Ainsi, (2) implique (3). Clairement (3) implique (4). Supposons

maintenant que (4) soit vrai : il existe un ensemble d'arcs qui forment des cycles élémentaires *disjoints* et dont le graphe de communication contient un arc allant de i vers j mais pas de chemin allant de j vers i. L'un de ces cycles élémentaires admet un arc allant de i vers j dans son graphe de communication. Ce graphe de communication ne peut pas avoir de chemin allant de j vers i. Ainsi, (4) implique (2). □

Soit (i, j) un canal. Nous allons construire une formule booléenne $\Phi_{\text{div}}(\mathcal{G}, i, j)$ avec deux types de variables : les arcs $a \in A$ et les instances $k \in \mathcal{I}$. Ainsi, il y a $|A| + |\mathcal{I}|$ variables. Pour tout arc $a \in A$ de q vers q', nous écrivons $s \rightarrow_a r$ si le MSC contenu dans le domaine q ou le codomaine q' de a contient un échange de messages allant de s vers r, c'est-à-dire s'il contient un événement étiqueté par $s!r$ ou $r?s$.

Les variables d'arc a affectées à vrai correspondront à un sous-ensemble d'arcs qui formeront un ensemble disjoint de cycles élémentaires. Pour ce faire, nous observons tout d'abord qu'un sous-ensemble d'arcs forme un ensemble de cycles élémentaires disjoints si, et seulement si, les trois propriétés suivantes sont satisfaites :

— un arc quitte un état si, et seulement si, un arc entre dans cet état,

— il y a au plus un arc sortant d'un état,

— il y a au plus un arc entrant dans un état.

Ces conditions peuvent être formalisées par les conditions booléennes suivantes :

$$\text{Cycles\_Elementaires\_Disjoints} = \bigwedge \begin{cases} \bigwedge\limits_{a \in A} \left(a \rightarrow \bigvee\limits_{dom(a')=cod(a)} a' \right) \\ \bigwedge\limits_{a \in A} \left(a \rightarrow \bigvee\limits_{cod(a')=dom(a)} a' \right) \\ \bigwedge\limits_{a \neq a' \text{ et } dom(a)=dom(a')} (\neg a \vee \neg a') \\ \bigwedge\limits_{a \neq a' \text{ et } cod(a)=cod(a')} (\neg a \vee \neg a') \end{cases}$$

Rappelons que $a \rightarrow b$ est équivalent à $\neg a \vee b$. Donc Cycles_Elementaires_Disjoints est presque en forme normale conjonctive (CNF), ce qui est nécessaire pour la plupart des SAT-solveurs.

Remarquons ici que nous considérons des cycles élémentaires disjoints. Cela peut sembler à priori plus compliqué que de considérer un cycle élémentaire. En réalité, écrire une formule booléenne sélectionnant un cycle élémentaire est bien plus complexe que de sélectionner des cycles élémentaires disjoints. La raison est que nous n'avons pas à vérifier qu'il n'y ait qu'un et un seul cycle élémentaire. Ainsi, avec quelques contraintes au niveau des nœuds nous pouvons facilement sélectionner des cycles élémentaires disjoints.

Nous considérons à présent le sous-ensemble d'instances assignées à vrai. Nous complétons la formule afin que toutes les instances accessibles depuis l'instance fixée j dans le graphe de communication des arcs sélectionnés soient affectées à vrai (mais possiblement d'autres instances également). En particulier, la variable j doit être attribuée à vrai. Cette

exigence peut être formalisée en CNF de la manière suivante :

$$\text{Reachable}_j = j \wedge \bigwedge_{s \to_a r} (a \wedge s) \to r$$

Ceci exprime le fait que si par exemple s est accessible à partir de j, et que a est sélectionné, alors r est accessible à partir de j, dès lors que le domaine ou codomaine de a porte un MSC contenant un échange de messages allant de s vers r. Rappelons maintenant que le canal (i, j) est divergent dans $\mathcal{G}$ s'il existe un ensemble de cycles élémentaires disjoints dans $\mathcal{G}$ dont le graphe de communication contient un arc allant de i vers j mais pas de chemin allant de j vers i (propriété 5.1.1), ce qui signifie que la variable i peut être assignée à faux. Cela se traduit par la condition suivante :

$$\text{Divergence}_{i,j} = (\neg i) \wedge \bigvee_{i \to_a j} a$$

Pour conclure, nous considérons la formule

$$\Phi_{\text{div}}(\mathcal{G}, i, j) = \text{Cycles\_Elementaires\_Disjoints} \wedge \text{Reachable}_j \wedge \text{Divergence}_{i,j}.$$

Théorème 5.1.2. *Le canal (i, j) est divergent dans $\mathcal{G}$ si, et seulement si, la formule booléenne $\Phi_{div}(\mathcal{G}, i, j)$ est satisfiable.*

Démonstration. Supposons que (i, j) soit divergent dans $\mathcal{G}$. Il existe un cycle élémentaire γ dont le graphe de communication contient un arc allant de i vers j mais pas de chemin allant de j vers i. Assignons à vrai tous les arcs de ce cycle et à faux tous les autres arcs. Assignons à vrai toutes les instances accessibles à partir de j dans le graphe de communication de γ, et à faux les autres instances. Une telle assignation satisfait $\Phi_{\text{div}}(\mathcal{G}, i, j)$.

Inversement, supposons que $\Phi_{\text{div}}(\mathcal{G}, i, j)$ est satisfiable et considérons une affectation satisfaisante. Le sous-ensemble d'arcs assignés à vrai forme un ensemble de cycles élémentaires disjoints. Par ailleurs l'ensemble des instances accessibles à partir j dans le graphe de communication de γ sont affectées à vrai, mais i est affecté à faux : par conséquent, il n'y a pas de chemin allant de j vers i. Pourtant, il y a un arc allant de i vers j grâce à $\text{Divergence}_{i,j}$. Il s'ensuit que (i, j) est divergent dans $\mathcal{G}$. □

Notons que le nombre de clauses dans Cycles_Elementaires_Disjoints est d'au plus $|A| + |A| + |A| \times (|A| - 1) + |A| \times (|A| - 1) = 2 \times |A|^2$. Le nombre de clauses dans Reachable_j est d'au plus $1 + |E|$ avec $|E|$ le nombre d'événements total contenu dans les nœuds du MSG. Il y a 2 clauses dans $\text{Divergence}_{i,j}$. Ainsi, le nombre de clauses dans $\Phi_{\text{div}}(\mathcal{G}, i, j)$ est d'au plus $2 \times |A|^2 + |E| + 3$. En outre, le nombre de littéraux dans chaque clause est d'au plus $1 + |A|$.

Fait intéressant, dans le cas où (i, j) est divergent alors toute assignation satisfaisant $\Phi_{\text{div}}(\mathcal{G}, i, j)$ fournit un contre-exemple pour la non-divergence de ce canal sous la forme d'un cycle élémentaire dont le graphe de communication contient un arc allant de i vers j mais pas de chemin allant de j vers i. Un tel cycle élémentaire peut être utile en pratique pour des outils afin de retourner un contre-exemple à l'utilisateur. Ainsi, afin de détecter

une divergence dans un MSG $\mathcal{G}$, nous avons simplement besoin d'utiliser un SAT-solveur et l'interroger sur la satisfiabilité de $\Phi_{\text{div}}(\mathcal{G}, i, j)$ pour tous les canaux (i, j). Nous pouvons également gagner du temps en nous limitant aux canaux qui sont effectivement utilisés dans $\mathcal{G}$.

Remarque 5.1.3. *La formule Cycles_Elementaires_Disjoints veille à ce que les arcs affectés à vrai forment des cycles élémentaires disjoints dans $\mathcal{G}$. Par conséquent, le codomaine de chacun de ces arcs est également le domaine d'un de ces arcs. Ainsi, afin d'éviter de la redondance, nous pouvons restreindre la conjonction dans $Reachable_j$ et la disjonction dans $Divergence_{i,j}$ aux MSC apparaissant dans les domaines de ces arcs. De plus, un cycle comprend un événement $s!r$ si, et seulement si, elle contient un événement étiqueté par $r?s$. En conséquence, nous pouvons rechercher des événements d'émission seulement. Une autre amélioration simple serait de calculer d'abord les composantes fortement connexes de $\mathcal{G}$ et vérifier chaque composante séparément. Ces trois remarques s'appliquent également pour la coopération globale dans la suite de ce chapitre.*

Remarque 5.1.4. *Afin de détecter la divergence d'un MSG $\mathcal{G}$, nous pouvons éviter de vérifier chaque canal séparément. Pour cela, nous pouvons fabriquer une formule $\Psi_{div}(\mathcal{G})$ qui est satisfiable si, et seulement si, $\Phi_{div}(\mathcal{G}, i, j)$ est satisfiable pour chaque canal (i, j). Pour ce faire, nous considérons $2 \times |\mathcal{I}|$ nouvelles variables $(i_k)_{k\in\mathcal{I}}$ et $(j_k)_{k\in\mathcal{I}}$. Nous exigeons qu'exactement une instance i_k (respectivement j_k) soit affectée à vrai. Intuitivement i_k est affecté à vrai si $k = i$ et j_k est affecté à vrai si $k = j$. Maintenant, $\Psi_{div}(\mathcal{G})$ est obtenue à partir de $\Phi_{div}(\mathcal{G}, i, j)$ en remplaçant la clause $\neg i$ par $\bigwedge_{k\in\mathcal{I}}(i_k \to \neg k)$, la clause j par $\bigwedge_{k\in\mathcal{I}}(j_k \to k)$, et la clause $\bigvee_{i\to_a j} a$ par $\bigwedge_{k,l\in\mathcal{I}}((i_k \wedge j_l) \to \bigvee_{k\to_a l} a)$.*

$$Divergence = (\bigwedge_{k\in\mathcal{I}} (i_k \to \neg k)) \wedge (\bigwedge_{k,l\in\mathcal{I}} ((i_k \wedge j_l) \to \bigvee_{k\to_a l} a))$$

$$Reachable = \bigwedge_{k\in\mathcal{I}} (j_k \to k) \wedge \bigwedge_{s\to_a r} (a \wedge s) \to r$$

$$\Psi_{div}(\mathcal{G}) = Cycles\_Elementaires\_Disjoints \wedge Reachable \wedge Divergence$$

5.2 Coopération globale avec SAT

De même que pour la propriété 5.1.1, nous voulons nous concentrer sur des cycles élémentaires (ou des ensembles des cycles élémentaires disjoints) pour vérifier la coopération globale d'un MSG à l'aide de SAT-solveurs. Cependant, il n'existe pas de propriétés analogues à la propriété 5.1.1 pour la coopération globale : considérons par exemple le MSG avec trois états décrit dans la figure 5.1 ; ce MSG n'est pas globalement coopératif, mais le graphe de communication de tout cycle *élémentaire* est connexe. Pour cette raison, nous adoptons dans la propriété suivante une contrainte sur les états étiquetés par un MSC vide.

Propriété 5.2.1. *Supposons qu'aucun état dans un cycle de $\mathcal{G}$ ne soit étiqueté par un MSC vide. Alors $\mathcal{G}$ est globalement coopératif si, et seulement si, le graphe de communication de*

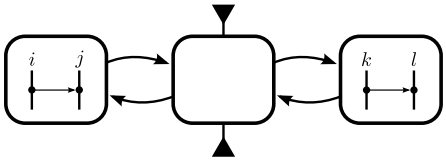


FIGURE 5.1 – MSG pour la propriété 5.2.1.

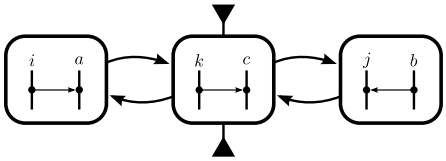


FIGURE 5.2 – MSG pour la remarque 5.2.2.

tout cycle élémentaire *est connexe.*

Démonstration. La condition est évidemment nécessaire. Nous supposons que le graphe de communication de tout cycle *élémentaire* est connexe. Nous procédons par contradiction et supposons que $\mathcal{G}$ n'est pas globalement coopératif. Alors, il existe un cycle $\gamma = q_0 \longrightarrow q_1 ... q_{n-1} \longrightarrow q_n = q_0$ dont le graphe de communication n'est pas connexe. On peut supposer que ce cycle est de longueur minimale. Comme ce cycle n'est pas élémentaire, il existe deux entiers naturels k et l tel que $0 \leq k < l < n$ et $q_k = q_l$. Soit i une instance active dans q_k. Alors

1. toutes les instances actives dans le cycle $q_k \longrightarrow q_{k+1} ... q_{l-1} \longrightarrow q_l$ sont connexes avec i dans son graphe de communication, car γ est de longueur minimale ;
2. toutes les instances actives dans le cycle $q_0 \longrightarrow q_1 ... q_k \longrightarrow q_{l+1} \longrightarrow q_{l+2} ... q_{n-1} \longrightarrow q_n$ sont connexes avec i dans son graphe de communication, car γ est de longueur minimale.

Il s'ensuit que toutes les instances actives dans le cycle $q_0 \longrightarrow q_1 ... q_{n-1} \longrightarrow q_n$ sont connexes avec i dans son graphe de communication : contradiction. □

Remarque 5.2.2. *Contrairement à la propriété 5.1.1, la propriété 5.2.1 échoue* si l'on considère un canal fixé. *Considérons par exemple le MSG représenté dans la figure 5.2. Le canal (i, j) n'est pas globalement coopératif, donc $\mathcal{G}$ n'est pas globalement coopératif. Cependant i et j sont connexes dans le graphe de communication de tout cycle élémentaire pour lequel à la fois i et j sont actifs (car il n'y a pas de tel cycle élémentaire). Pourtant, le résultat ci-dessus affirme qu'il existe un cycle* élémentaire *γ et deux instances actives dans γ qui apparaissent dans des composantes connexes distinctes du graphe de communication de γ. Cela est vérifié par exemple avec i et k.*

Notez ici que nous pouvons facilement vérifier si la contrainte additionnelle sur les états vides est satisfaite par $\mathcal{G}$ en deux étapes. Nous calculons les composantes fortement connexes de $\mathcal{G}$ considéré comme un graphe orienté, puis nous vérifions les deux conditions suivantes :

— chaque état à l'intérieur d'une composante d'au moins deux états n'est pas étiqueté par un MSC vide ;

— chaque état q muni d'un arc $q \to q$ bouclant sur lui même n'est pas étiqueté par un MSC vide.

Il est facile de retirer du MSG $\mathcal{G}$ tous les états étiquetés par un MSC vide qui apparaissent dans un cycle afin d'obtenir un MSG équivalent : cette suppression des états vides préserve et reflète la coopération globale.

Propriété 5.2.3. *Soit $\mathcal{G}$ un MSG et q_0 un état étiqueté par un MSC vide et qui apparaît dans un cycle. Soit $\mathcal{G}'$ le MSG obtenu à partir de $\mathcal{G}$ par*

1. *l'ajout d'un arc $q_1 \to q_2$ dans $\mathcal{G}'$ chaque fois qu'il y a deux arcs $q_1 \to q_0$ et $q_0 \to q_2$ dans $\mathcal{G}$;*
2. *puis la suppression de tous les arcs de $\mathcal{G}$ ayant pour codomaine q_0.*

Alors $\mathcal{G}$ est globalement coopératif si, et seulement si, $\mathcal{G}'$ est globalement coopératif. Par ailleurs $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}')$.

Démonstration. Il est clair que pour chaque chemin γ réalisant un MSC $M@\chi$ dans $\mathcal{G}$, il existe un chemin γ' réalisant $M@\chi$ dans $\mathcal{G}'$ et inversement pour chaque chemin γ' réalisant un MSC $M@\chi$ dans $\mathcal{G}'$ il existe un chemin γ réalisant $M@\chi$ dans $\mathcal{G}$. Cette correspondance est valable pour les cycles. □

Nous pouvons itérer cette transformation à partir de tout MSG $\mathcal{G}$ et obtenir un nouveau MSG $\mathcal{G}'$ satisfaisant les trois contraintes suivantes :

1. $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}')$;
2. $\mathcal{G}$ est globalement coopératif si, et seulement si, $\mathcal{G}'$ est globalement coopératif ;
3. aucun état présent dans un cycle de $\mathcal{G}'$ n'est étiqueté par un MSC vide.

Nous supposons dans la suite qu'aucun état se produisant dans un cycle de $\mathcal{G}$ n'est étiqueté par un MSC vide. Afin de vérifier que $\mathcal{G}$ est globalement coopératif, nous fixons deux instances i et j et vérifions que pour tout cycle *élémentaire* dans $\mathcal{G}$, si i et j sont tous les deux actifs, alors ils sont connexes (propriété 5.2.1). Nous construisons une formule booléenne $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$ qui est satisfiable si, et seulement si, il existe un cycle élémentaire pour lequel i et j sont actifs, mais non connexes. La formule $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$ utilise $|A| + |\mathcal{I}| + |Q|$ des variables. Chaque arc $a \in A$, chaque instance $k \in \mathcal{I}$ et chaque état $q \in Q$ correspond à une variable. La formule $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$ se compose de quatre parties et emprunte la spécification de Cycles_Elementaires_Disjoints :

$$\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j) \quad = \quad \text{Cycles\_Elementaires\_Disjoints} \wedge \text{Unconnected}_{i,j} \wedge \text{Init}_i \wedge \text{Propagation}_{i,j}$$

La formule Cycles_Elementaires_Disjoints nous garantit ici encore que les arcs a assignés à vrai forment une collection de cycles élémentaires disjoints.

$$\text{Unconnected}_{i,j} = \bigwedge \begin{cases} \bigvee_{a:q\to q' \text{ tel que } i\in active(q)} a \\ i \\ \bigwedge_{s\to_a r}(a \wedge s) \to r \\ \bigwedge_{s\to_a r}(a \wedge r) \to s \\ \neg j \end{cases}$$

La première clause garantit que l'instance i est active dans le graphe de communication des cycles disjoints sélectionnés. De plus, les instances connectées à i dans le graphe de communication des arcs sélectionnés sont toutes assignées à vrai et j est assigné à faux. Ainsi, les formules Cycles_Elementaires_Disjoints et $\text{Unconnected}_{i,j}$ sélectionnent des cycles élémentaires disjoints tels que j n'apparaît pas dans la composante connexe de i.

Le reste de la formule va garantir que les instances i et j sont actives simultanément dans l'un des cycles élémentaires sélectionnés.

$$\text{Init}_i = \bigvee_{a:q\to q' \text{ tel que } i\in active(q)}(a \wedge q)$$

Si cette formule est satisfaite, alors premièrement au moins un état est affecté à vrai et dans cet état i est actif. Deuxièmement, cet état se trouve sur un cycle sélectionné. Bien que cette formule soit simple, nous préférerons la formule suivante qui s'écrit plus facilement sous forme normale conjonctive.

$$\text{Init}_i = \bigwedge \begin{cases} \bigvee_{q \text{ tel que } i\in active(q)} q \\ \bigwedge_{q \text{ tel que } i\in active(q)}(q \to \bigvee_{a:q\to q'} a) \end{cases}$$

Cette formule garantit également qu'au moins un état est affecté à vrai, que dans cet état i est actif et que cet état se trouve sur un cycle sélectionné. Elle exige en outre que tout état affecté à vrai dans lequel i est actif fait partie des cycles sélectionnées.

Nous terminons par la formule $\text{Propagation}_{i,j}$ qui garantit que chaque cycle élémentaire sélectionné, contenant un état q dans lequel i actif, contient un état q' dans lequel j est actif.

$$\text{Propagation}_{i,j} = \bigwedge \begin{cases} \bigwedge_{a:q\to q' \text{ tel que } j\notin q'}(a \wedge q \to q') \\ \bigwedge_{a:q\to q' \text{ tel que } i\in q' \text{ et } j\notin q'} \neg(q \wedge a) \end{cases}$$

Théorème 5.2.4. *Soit $\mathcal{G}$ un MSG dans lequel aucun cycle ne contient un état étiqueté par le MSC vide. Alors $\mathcal{G}$ n'est pas globalement coopératif si, et seulement si, la formule booléenne $\Phi_{\overline{gc}}(\mathcal{G}, i, j)$ est satisfiable pour un canal (i, j).*

Démonstration. Supposons que la formule $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$ soit satisfaite pour une assignation. Par la formule Init_i, un état q contenant une instance i active est assigné à vrai et un arc a sortant de q est assigné à vrai. Par la formule Cycles_Elementaires_Disjoints, tous les arcs assignés à vrai forment des cycles élémentaires disjoints. Par conséquent l'arc a est contenu dans un cycle élémentaire γ disjoint des autres cycles. La première partie de la formule $\text{Propagation}_{i,j}$, contraint à assigner à vrai les états le long du cycle γ en commençant par q et jusqu'à rencontrer éventuellement un état pour lequel j est actif. De plus, la formule

Propagation$_{i,j}$ contraint à assigner à faux tous les états de γ qui précèdent un état pour lequel i est actif et j n'est pas actif. Ainsi, si q ne contient pas l'instance j comme instance active, alors les états de γ ne peuvent pas tous être assignés à vrai. Cela est possible que s'il existe un état pour lequel l'instance j est active dans γ. Si q contient l'instance j active, alors γ contient aussi un état pour lequel l'instance j est active. Ainsi, le graphe de communication formé par γ contient les instances i et j. Par la formule Unconnected$_{i,j}$, les instances i et j sont chacune dans une composante connexe distincte dans le graphe de communication formé par γ. Ainsi, $\mathcal{G}$ n'est pas globalement coopératif.

Supposons maintenant que $\mathcal{G}$ ne soit pas globalement coopératif. Comme aucun état dans un cycle ne contient de MSC vide, il existe un cycle *élémentaire* γ dont le graphe de communication n'est pas connexe (propriété 5.2.1). Soit i et j deux instances actives présentent dans deux composantes connexes différentes. Nous montrons que l'on peut satisfaire la formule $\Phi_{\overline{gc}}(\mathcal{G}, i, j)$. Nous assignons à vrai les variables correspondant aux arcs qui constituent γ, et à faux les autres arcs. Il est clair que Cycles_Elementaires_Disjoints est alors satisfaite. Ensuite, deux cas sont possibles :

1. Tous les états de γ contenant une instance i active, contiennent également une instance j active. Il suffit alors d'assigner à vrai tous les états de γ et assigner à faux tous les autres états.
2. Il existe un état q avec i actif dans lequel j n'est pas actif. Ainsi, il existe un autre état q' dans lequel j est actif sur le cycle γ. Nous pouvons supposer que q et q' sont tels qu'il n'existe pas, sur le chemin allant de q à q', d'état dans lequel i est actif et j n'est pas actif. Ainsi, les états le long du chemin allant de q à q' dans γ peuvent être :
   - des états avec i et j non actifs ;
   - des états avec i et j actifs ;
   - des états avec i non actif et j actif.

   Nous choisissons d'assigner à vrai tous les états de q (inclus) à q' (exclus) le long de γ et à faux tous les autres états. Ainsi Init$_i$ est satisfaite. On vérifie aussi aisément que Propagation$_{i,j}$ est satisfaite.

□

On obtient de cette façon une assignation qui satisfait $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$.

Rappelons que le nombre maximal de clauses dans Cycles_Elementaires_Disjoints est d'au plus $2\times|A|^2$. Le nombre maximal de clauses dans Unconnected$_{i,j}$ est d'au plus $3+2\times|A|$. Le nombre maximal de clauses dans Init$_i$ est d'au plus $1 + |E|$ avec E le nombre total d'événements dans le MSG. Le nombre maximal de clauses dans Propagation$_{i,j}$ est d'au plus $2 \times |A|$. Ainsi il y a au plus $2 \times |A|^2 + 3 \times |A| + E + 4$ clauses dans $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$. Ces clauses ont au plus $|A| + 1$ littéraux.

Remarque 5.2.5. *Comme pour la remarque 5.1.4, nous pouvons éviter la vérification de chaque canal séparément à l'aide d'une formule unique $\Psi_{\overline{gc}}(\mathcal{G})$ qui est satisfiable si, et seulement si, $\Phi_{\overline{gc}}(\mathcal{G}, i, j)$ est satisfiable pour un canal (i, j), c'est-à-dire si $\mathcal{G}$ n'est pas globalement coopératif. Pour ce faire, nous considérons $2 \times |\mathcal{I}|$ nouvelles variables $(i_k)_{k\in\mathcal{I}}$ et $(j_k)_{k\in\mathcal{I}}$.*

Nous exigeons qu'exactement une instance i_k (respectivement j_k) soit affectée à vrai. Intuitivement i_k est affecté à vrai si $k = i$ et j_k est affecté à vrai si $k = j$. Ci-dessous, une adaptation de Init, Propagation et Unconnected.

$$Init = \bigwedge \begin{cases} \bigwedge_{k\in\mathcal{I}} i_k \rightarrow (\bigvee_{q \text{ tel que } k\in active(q)} q) \\ \bigwedge_{k\in\mathcal{I}} (\bigwedge_{q \text{ tel que } k\in active(q)} ((i_k \wedge q) \rightarrow \bigvee_{a:q\rightarrow q'} a)) \end{cases}$$

$$Propagation = \bigwedge \begin{cases} \bigwedge_{l\in\mathcal{I}} (\bigwedge_{a:q\rightarrow q' \text{ tel que } l\notin active(q')} (j_l \wedge a \wedge q \rightarrow q')) \\ \bigwedge_{k,l\in\mathcal{I}} (\bigwedge_{a:q\rightarrow q' \text{ tel que } k\in active(q') \text{ et } l\notin active(q')} j_l \rightarrow \neg(q \wedge a)) \end{cases}$$

$$Unconnected = \bigwedge \begin{cases} \bigwedge_{s\rightarrow_a r} (a \wedge s) \rightarrow r \\ \bigwedge_{s\rightarrow_a r} (a \wedge r) \rightarrow s \\ \bigwedge_{k\in\mathcal{I}} i_k \rightarrow (\bigvee_{a:q\rightarrow q' \text{ tel que } k\in active(q)} a) \\ \bigwedge_{l\in\mathcal{I}} (j_k \rightarrow \neg k) \end{cases}$$

5.3 Accessibilité par préfixe avec SAT

Le problème de l'accessibilité par préfixe pour un MSG *non-divergent* $\mathcal{G}$ et un marquage χ' consiste à vérifier si χ' est accessible par préfixe dans $\mathcal{G}$, c'est-à-dire s'il existe un MSC accepté $M@\chi$ et un préfixe $M'@\chi$ de $M@\chi$ dont la cible est χ'.

Considérons un chemin $q_1 \rightarrow ... \rightarrow q_n$ dans $\mathcal{G}$ et un préfixe $M'@\chi$ du MSC produit $M@\chi = \mu(q_1)\cdot ...\cdot \mu(q_n)$. Alors la cible de $M'@\chi$ est accessible par préfixe. La raison est que pour tout chemin $q_1{}^\circ \rightarrow ... \rightarrow q_m{}^\circ$ avec $q_1{}^\circ = \imath$ et $q_m{}^\circ = q_1$, la cible de $M'@\chi$ est la cible de $M^\circ@\chi^\circ \cdot M'@\chi$ où $M^\circ@\chi^\circ = \mu(q_1{}^\circ)\cdot ...\cdot \mu(q_m{}^\circ)$. De plus $M^\circ@\chi^\circ \cdot M'@\chi$ est un préfixe de $M^\circ@\chi^\circ \cdot M@\chi$ et ce produit appartient à $\mathcal{L}(\mathcal{G})$. Nous pouvons alors supposer qu'au moins un événement de $\mu(q_1)$ n'appartient pas à M' et qu'au moins un événement de $\mu(q_n)$ lui appartient. La propriété inverse est vraie :

Propriété 5.3.1. *Soit $\mathcal{G}$ un MSG non-divergent. Soit χ' un marquage. Si χ' est préfixe-accessible alors il y a un chemin $q_1 \rightarrow ... \rightarrow q_n$ tel que χ' est la cible d'un préfixe du MSC produit $\mu(q_1)\cdot ...\cdot \mu(q_n)$ pour lequel*

- *un événement de $\mu(q_1)$ n'appartient pas au préfixe,*
- *un événement de $\mu(q_n)$ appartient au préfixe.*

Démonstration. Il existe un MSC $M@\chi = (E, \preceq, \xi, \chi)$ dans le langage de MSC $\mathcal{L}(\mathcal{G})$, une extension linéaire $s = (E, \leq, \xi)$ de M, et un préfixe $s' \leq s$ tel que $\chi(i,j) + \#^{i!j}(s') - \#^{j?i}(s') = \chi'(i,j)$ pour chaque canal $(i,j) \in \mathcal{K}$. Le MSC $M@\chi$ est le produit de MSC portés par des états successifs de $\mathcal{G}$ le long d'un chemin $\imath = q_1 \rightarrow ... \rightarrow q_n$ qui débute de l'état initial $\imath$. Nous posons $M_p@\chi_p = \mu(q_p)$ pour chaque $p \leq n$. Alors $M@\chi = M_1@\chi_1 \cdot ... \cdot M_n@\chi_n$ et $\chi = \chi_1$. Les événements de s' correspondent à des préfixes M'_p de M_p. Soit m le premier indice tel qu'un événement dans M_m n'appartient pas à M'_m. Soit m' le dernier indice tel qu'un événement dans $M_{m'}$ apparaît dans $M'_{m'}$, c'est-à-dire $M'_{m'}$ n'est pas vide. Ainsi, $M'_p = M_p$ pour tout $p < m$ et M'_p est vide pour tout $p > m'$. Nous considérons le chemin

$q_m \to ... \to q_{m'}$. Puisque tous les états après $q_{m'}$ sont inutiles pour le préfixe que nous considérons, nous pouvons supposer que $m' = n$. Soit M'' le sous-ensemble d'événements de M' qui n'appartient pas au MSC $M_1@\chi_1 \cdot ... \cdot M_{m-1}@\chi_{m-1}$. Alors, le MSC $M''@\chi_m$ est un préfixe du MSC $M_m@\chi_m \cdot ... \cdot M_n@\chi_n$. De plus, sa cible est égale à χ'. □

Le résultat suivant montre que nous pouvons limiter la longueur des chemins à considérer.

Propriété 5.3.2. *Soit $\mathcal{G}$ un MSG non-divergent. Soit*

$$F = \max_{q \in Q} \max_{\mathcal{I}' \subseteq \mathcal{I}} \sum_{i \notin \mathcal{I}' \text{ et } j \in \mathcal{I}'} source(\mu(q))(j,i)$$

où $source(\mu(q))$ désigne la source du marquage du MSC $\mu(q)$. Un marquage χ' est préfixe-accessible si, et seulement si, il y a un chemin $q_1 \to ... \to q_n$ de longueur $n \leq (1+F)\times|Q|\times|\mathcal{I}|$ tel que χ' est la cible d'un préfixe du MSC de produit $\mu(q_1) \cdot ... \cdot \mu(q_n)$.

Démonstration. Soit χ' un marquage préfixe-accessible dans $\mathcal{G}$. Par la propriété 5.3.1, il y a un chemin $q_1 \to ... \to q_n$ tel que

- χ' est la cible d'un préfixe $M'@\chi$ du MSC produit $M@\chi = \mu(q_1) \cdot ... \cdot \mu(q_n)$
- un événement de $\mu(q_1)$ n'appartient pas au préfixe $M'@\chi$,
- un événement de $\mu(q_n)$ appartient au préfixe $M'@\chi$.

Nous supposons que la longueur n du chemin $q_1 \to ... \to q_n$ est *minimale* parmi tous les chemins qui satisfont ces trois propriétés. Il reste à montrer que $n \leq (1 + F) \times |Q| \times |\mathcal{I}|$. Pour chaque $p \leq n$, nous posons $M_p@\chi_p = \mu(q_p)$ et nous considérons le préfixe M'_p de M_p qui rassemble tous les événements de M_p qui se produisent dans M'. Alors $M'@\chi = M'_1@\chi_1 \cdot ... \cdot M'_n@\chi_n$. Soit $J_p \subseteq \mathcal{I}$ le sous-ensemble des instances k telles qu'un événement dans $M_1@\chi_1 \cdot ... \cdot M_p@\chi_p$ survenant sur k n'appartient pas au préfixe $M'@\chi$. Clairement, nous avons $J_p \subseteq J_{p+1}$ pour tout p. En outre, il existe des instances $j \in \mathcal{I}$ telles que $j \in J_1$ car des événements de M_1 n'appartiennent pas à M'_1. Ainsi, $J_1 \neq \emptyset$ et donc il y a au plus $|\mathcal{I}|$ ensembles distincts J_p.

Nous procédons maintenant par contradiction et supposons que $n \geq (1+F)\times|Q|\times|\mathcal{I}|+1$. Il y a deux entiers naturels p et l tels que $1 \leq p < p+l \leq n$, $J_p = J_{p+l}$ et $l+1 > (1+F)\times|Q|$. Autrement il y aurait au plus $(1 + F) \times |Q|$ états q_p avec le même ensemble J_p et donc il y aurait au plus $(1+F)\times|Q|\times|\mathcal{I}|$ états dans le chemin $q_1 \to ... \to q_n$, i.e. $n \leq (1+F)\times|Q|\times|\mathcal{I}|$. Parmi les événements qui se produisent dans M'_{p+1}, ..., M'_{p+l} le nombre d'événements de réception $i?j$ avec $i \notin J_p$ et $j \in J_p$ est au plus égal à $\sum_{i \notin J_p \text{ et } j \in J_p} \chi_{p+1}(j,i)$, par conséquent, il est au plus égal à F. La raison est que $\chi'_{p+1}(j,i) \leq \chi_{p+1}(j,i)$ et aucun événement ne se produit sur j dans les MSC M'_{p+1}, ..., M'_{p+l} puisque $j \in J_p$. Comme $l \geq (1 + F) \times |Q|$ le chemin q_{p+1}, ..., q_{p+l} contient un cycle q_r, ..., $q_{r+k} = q_r$ (avec $k \geq 1$) tel qu'aucune réception sur $\mathcal{I} \setminus J_p$ de messages provenant de J_p ne se produit dans le MSC $\mu(q_r) \cdot ... \cdot \mu(q_{r+k})$. Comme $\mathcal{G}$ n'est pas divergent, aucun événement d'émission de $\mathcal{I} \setminus J_p$ vers J_p ne se produit dans le MSC produit $\mu(q_r) \cdot ... \cdot \mu(q_{r+k})$ non plus. Nous expliquons ci-dessous pourquoi nous pouvons éliminer ce cycle du chemin q_1, ..., q_n tout en préservant ses trois propriétés. Autrement dit,

il existe un préfixe du MSC $M_1@\chi_1 \cdot ... \cdot M_r@\chi_r \cdot M_{r+k+1}@\chi_{r+k+1} \cdot ... \cdot M_n@\chi_n$ dont la cible est aussi χ'. Cela contredit la minimalité de n. Ainsi, $n \leq (1 + F) \times |Q| \times |\mathcal{I}|$.

Nous commençons l'explication annoncée par trois observations :

1. Pour chaque $m \in [r+1..r+k]$, M'_m est précisément la restriction de M_m aux événements qui se produisent sur les instances de $\mathcal{I} \setminus J_p$ car $J_p = J_r = J_{r+k}$.
2. Comme $q_r = q_{r+k}$, la cible de $M_{r+k}@\chi_{r+k}$ est la cible de $M_r@\chi_r$, c'est-à-dire χ_{r+1}. Ainsi, la cible du produit de MSC $M_{r+1}@\chi_{r+1} \cdot ... \cdot M_{r+k}@\chi_{r+k}$ est égale à χ_{r+1}.
3. Le préfixe $M'@\chi$ de $M@\chi$ est le produit de MSC $M'@\chi = M'_1@\chi'_1 \cdot ... \cdot M'_n@\chi'_n$. En outre, le marquage cible de $M'_n@\chi'_n$ est égal à la cible de $M'@\chi$, qui est, χ'.

Nous savons qu'aucun événement étiqueté par $i?j$ avec $j \in J_p$ et $i \in \mathcal{I} \setminus J_p$ ne se produit dans le produit de MSC $M_{r+1}@\chi_{r+1} \cdot ... \cdot M_{r+k}@\chi_{r+k}$. Par conséquent, il n'existe aucun arc de J_p à $\mathcal{I} \setminus J_p$ dans le graphe de communication du cycle q_{r+1}, ..., $q_{r+k} = q_r$. Comme $\mathcal{G}$ est non-divergent, il n'y a aucun arc de $\mathcal{I} \setminus J_p$ à J_p dans ce graphe de communication. Par conséquent, le MSC produit $M'_{r+1}@\chi'_{r+1} \cdot ... \cdot M'_{r+k}@\chi'_{r+k}$ ne contient pas d'échange de messages entre J_p et $\mathcal{I} \setminus J_p$. Il s'ensuit que la cible de $M'_{r+1}@\chi'_{r+1} \cdot ... \cdot M'_{r+k}@\chi'_{r+k}$ est égale à χ'_{r+1}. Plus précisément, nous avons

— $\chi'_{r+k}(k,l) = \chi'_{r+1}(k,l)$ si $k \in J_p$ et $l \in J_p$ car rien ne se passe sur J_p et donc sur ce canal.

— $\chi'_{r+k}(k,l) = \chi'_{r+1}(k,l)$ si $k \in \mathcal{I} \setminus J_p$ et $l \in \mathcal{I} \setminus J_p$ car la cible de $M'_{r+1}@\chi'_{r+1} \cdot ... \cdot M'_{r+k}@\chi'_{r+k}$ est égal à sa source, car il correspond à un cycle.

— $\chi'_{r+k}(k,l) = \chi'_{r+1}(k,l)$ si $k \in J_p$ et $l \in \mathcal{I} \setminus J_p$ car aucun événement n'agit sur ce canal.

— $\chi'_{r+k}(k,l) = \chi'_{r+1}(k,l)$ si $k \in \mathcal{I} \setminus J_p$ si $l \in J_p$ car il n'y a pas d'événement sur ce canal.

Par conséquent, le MSC produit $M''@\chi = M'_1@\chi'_1 \cdot ... \cdot M'_r@\chi'_r \cdot M'_{r+k+1}@\chi'_{r+k+1} \cdot ... \cdot M'_n@\chi'_n$ est valide. Ce MSC produit est un préfixe du MSC produit $M_1@\chi_1 \cdot ... \cdot M_r@\chi_r \cdot M_{r+k+1}@\chi_{r+k+1} \cdot ... \cdot M_n@\chi_n$. De plus la cible de $M''@\chi$ est égal à la cible de $M'_n@\chi'_n$, qui est χ'. □

Ainsi, nous devons explorer seulement des chemins de longueur bornée dans $\mathcal{G}$. Comme première conséquence, nous obtenons la correspondance annoncée entre non-divergence et canal borné qui généralise [18, Th. 5].

Corollaire 5.3.3. *Un MSG $\mathcal{G}$ est à canaux bornés si, et seulement si, il n'est pas divergent.*

Démonstration. La propriété 5.3.2 montre que si $\mathcal{G}$ n'est pas divergent alors il est à canal borné. Supposons maintenant que (i,j) est un canal divergent dans $\mathcal{G}$. Il existe un chemin partant de l'état initial de $\mathcal{G}$, étiqueté par $M@\chi$, qui conduit à un cycle étiqueté par $N@\chi_0$ dont le graphe communication contient un arc de i vers j mais pas de chemin de j vers i. Par induction sur k, nous allons voir ci-dessous que $W_{i,j}(M@\chi \cdot (N@\chi_0)^r) \geq r$ pour tout $r \geq 0$, et par conséquent le canal (i,j) n'est pas borné dans $\mathcal{G}$.

Le cas où $r = 0$ est trivial. Soit J l'ensemble d'instances accessibles depuis l'instance j dans le graphe de communication de $N@\chi_0$. Nous avons $i \notin J$. Nous savons qu'il n'y a pas d'arc de J vers $\mathcal{I} \setminus J$ dans le graphe de communication de $N@\chi_0$. Soit $N'@\chi_0$ le préfixe de

N qui rassemble tous les événements survenus sur les instances de $\mathcal{I} \setminus J$. Alors la cible χ_1 du MSC $N'@\chi_0$ est telle que $\chi_1 \geq \chi_0$. Plus précisément, nous avons

- $\chi_1(k,l) = \chi_0(k,l)$ si $k \in J$ et $l \in J$ car rien ne se passe dans N' sur ce canal.
- $\chi_1(k,l) = \chi_0(k,l)$ si $k \in \mathcal{I} \setminus J$ et $l \in \mathcal{I} \setminus J$ car la cible de N est égale à sa source χ_0 (car N correspond à un cycle) et toutes les actions N sur ce canal se produisent dans N'.
- $\chi_1(k,l) = \chi_0(k,l)$ si $k \in J$ et $l \in \mathcal{I} \setminus J$ car aucun événement sur N n'agit sur ce canal.
- $\chi_1(k,l) \geq \chi_0(k,l)$ si $k \in \mathcal{I} \setminus J$ et $l \in J$ car tous les événements d'émission dans ce canal N apparaissent dans N'. En particulier, $\chi_1(i,j) > \chi_0(i,j)$ puisqu'aucune réception sur j n'apparaît dans N'.

Nous considérons le marquage $\delta = \chi_1 - \chi_0$ et définissons récursivement la séquence de marquage χ_r pour $r \geq 2$ par $\chi_{r+1} = \chi_r + \delta$. Alors, le MSC produit $M@\chi \cdot N'@\chi_0 \cdot \ldots \cdot N'@\chi_r$ est valide et sa cible est égale à χ_{r+1} avec $\chi_{r+1}(i,j) \geq r+1$. D'autre part, c'est un préfixe de $M@\chi \cdot (N@\chi_0)^{r+1}$. □

Le corollaire ci-dessous donne une borne supérieure pour le nombre de messages en attente dans un MSG non divergent.

Corollaire 5.3.4. *Soit $\mathcal{G}$ un MSG non divergent et (i,j) un canal donné. Soit $m_{i,j} = \max_{q \in Q} \#^{i!j}(\mu(q))$ le nombre maximal de messages envoyés allant de i vers j dans les MSC étiquetant un état de $\mathcal{G}$. Alors $W_{i,j}(\mathcal{G}) \leq m_{i,j} \times (1+F) \times |Q| \times |\mathcal{I}|$ où F est défini dans la propriété 5.3.2.*

Démonstration. La propriété 5.3.2 montre qu'il est suffisant de vérifier la largeur de canaux de MSC décrits par une séquence d'états de longueur au plus $(1+F) \times |Q| \times |\mathcal{I}|$. □

Remarque 5.3.5. *Il faut noter ici qu'il est nécessaire de connaître effectivement une borne pour $W_{i,j}(\mathcal{G})$ pour écrire la formule permettant de vérifier qu'un marquage est accessible par préfixe. La borne supérieure du corollaire précédant peut alors être utilisée. Néanmoins le calcul de F repose sur une recherche exponentielle dans le nombre de processus. Si cette recherche s'avère prohibitive, il est possible d'utiliser une borne moins fine. En effet,*

$$F \leq \max_{q \in Q} \sum_{(i,j) \in \mathcal{K}} \mathit{source}(\mu(q))(i,j)$$

.

Par souci de simplicité, nous supposerons dans la suite de ce chapitre qu'il y a au plus un événement dans chaque MSC porté par les états de $\mathcal{G}$. Notons que cette simplification n'est pas restrictive. En effet, le formalisme que nous adoptons est équivalent aux MSG compositionnels. Ainsi tout MSC peut se décomposer en une séquence de MSC contenant un seul événement.

La formule $\Phi_{\mathrm{p}}(\mathcal{G})$ utilise $H = (1+F) \times |Q| \times |\mathcal{I}|$ lignes de $|Q| + |\mathcal{I}| + \sum_{(i,j) \in \mathcal{K}} \lceil \log_2(W_{i,j}(\mathcal{G}) + 1) \rceil$ variables booléennes qui décrivent un chemin de longueur au plus H débutant de n'importe quel état. Dans chaque ligne x, ou de manière équivalente, à chaque étape de ce

chemin, les $|Q|$ premières variables $N_{a,x}$ (avec $1 \leq a \leq |Q|$) précisent quel état est atteint par le chemin à chaque étape : au plus une de ces variables peut être affectée à vrai. Les $|\mathcal{I}|$ variables suivantes $I_{p,x}$ (avec $p \in \mathcal{I}$) garde une trace des instances bloquées à un certain stade. Pour terminer, $\lceil \log_2(W_{i,j}(\mathcal{G}) + 1) \rceil$ variables $C_{b,i,j,x}$ sont utilisées pour chaque canal $i \to j$ (avec $1 \leq b \leq \lceil \log_2(W_{i,j}(\mathcal{G}) + 1) \rceil$). Comme d'après le corollaire 5.3.4, $W_{i,j}(\mathcal{G}) \leq (1 + F) \times |Q| \times |\mathcal{I}|$ (car $m_{i,j} \leq 1$), le nombre total de variables sur une ligne est inférieur à $|Q| + |\mathcal{I}| + |\mathcal{I}|^2 \times \lceil \log_2((1 + F) \times |Q| \times |\mathcal{I}| + 1) \rceil$.

Nous noterons $C_{i,j,x}$ le nombre binaire $\sum_{1 \leq b \leq \lceil \log_2(W_{i,j}(\mathcal{G})+1) \rceil} C_{b,i,j,x} \times 2^b$ représentant le nombre de messages en attente dans le canal (i, j) à l'étape x. Par souci de lisibilité, nous représenterons par $C_{i,j,x} + 1$ la formule requérant que $C_{i,j,x+1} = C_{i,j,x} + 1$ et par $C_{i,j,x} - 1$ la formule requérant que $C_{i,j,x+1} = C_{i,j,x} - 1$. Ces formules peuvent facilement s'écrire avec des formules booléennes, voici un exemple si les nombres sont codés sur trois bits :

$$C_{i,j,x} + 1 = \bigwedge \begin{cases} \neg C_{0,i,j,x} \to C_{0,i,j,x+1} \\ (\neg C_{1,i,j,x} \wedge C_{0,i,j,x}) \to (C_{1,i,j,x+1} \wedge \neg C_{0,i,j,x+1}) \\ (\neg C_{2,i,j,x} \wedge C_{1,i,j,x} \wedge C_{0,i,j,x}) \to (C_{2,i,j,x+1} \wedge \neg C_{1,i,j,x+1} \wedge \neg C_{0,i,j,x+1}) \\ (C_{2,i,j,x} \wedge C_{1,i,j,x} \wedge C_{0,i,j,x}) \to false \end{cases}$$

$$C_{i,j,x} - 1 = \bigwedge \begin{cases} C_{0,i,j,x} \to \neg C_{0,i,j,x+1} \\ (C_{1,i,j,x} \wedge \neg C_{0,i,j,x}) \to (\neg C_{1,i,j,x+1} \wedge C_{0,i,j,x+1}) \\ (C_{2,i,j,x} \wedge \neg C_{1,i,j,x} \wedge \neg C_{0,i,j,x}) \to (\neg C_{2,i,j,x+1} \wedge C_{1,i,j,x+1} \wedge C_{0,i,j,x+1}) \\ (\neg C_{2,i,j,x} \wedge \neg C_{1,i,j,x} \wedge \neg C_{0,i,j,x}) \to false \end{cases}$$

Nous commençons par écrire une formule exigeant que les variables $N_{a,x}$ sélectionnent un chemin de longueur au plus H dans le MSG. Ces conditions sont formalisées comme ceci :

$$\bigwedge_{1 \leq x \leq H, 1 \leq a < b \leq |Q|} (N_{a,x} \to \neg N_{b,x})$$

$$\bigwedge_{1 < x \leq H, 1 \leq a \leq |Q|} \left(N_{a,x} \to \bigvee_{b \in \mathrm{pred}(a)} N_{b,x-1} \right)$$

avec $b \in \mathrm{pred}(a)$ signifiant que b est un prédécesseur de a.

Nous devons maintenant représenter les échanges de message entre les différentes instances. Pour cela deux choses doivent être mises en place. Premièrement, nous devons compter le nombre de messages en attente dans chaque canal. Deuxièmement, nous devons être capable de bloquer une instance afin qu'elle ne puisse plus envoyer ni recevoir de message ; cela revient intuitivement à décider à quel moment la coupe du préfixe stoppe les instances.

Le blocage des processus est modélisé par les variables $I_{p,x}$ avec $p \in \mathcal{I}$. Si $I_{p,x}$ est assigné à faux, alors le processus p est bloqué à l'étape x et ne pourra plus envoyer ni recevoir de message. Si une instance est bloquée à une étape, alors elle reste bloquée par la suite :

$$\bigwedge_{p\in\mathcal{I},1\leq x\leq H} \neg I_{p,x} \rightarrow \neg I_{p,x+1}$$

Nous représentons ensuite les échanges de messages entre les différentes instances. Nous commençons par initialiser les compteurs à 0 et propageons leur valeurs lorsque aucun état n'est franchi.

$$\bigwedge \begin{cases} \bigwedge\limits_{i,j\in\mathcal{I}} C_{i,j,1} = 0 \\ \bigwedge\limits_{1<x\leq H} \left(\left(\bigwedge\limits_{1\leq a\leq |Q|} \neg N_{a_x} \right) \rightarrow (C_{i,j,x} = C_{i,j,x-1}) \right) \end{cases}$$

Si au contraire un nouvel état est ajouté au chemin, deux choix s'offrent à nous, soit nous effectuons l'action (réception ou émission), soit nous décidons que cette action ne sera pas effectuée dans le préfixe et nous bloquons le processus.

$$\bigwedge_{1\leq x\leq H,1\leq a\leq |Q|} N_{a,x} \rightarrow \begin{cases} (C_{i,j,x} + 1 \wedge I_{i,x}) \vee \neg I_{i,x} & \text{si } Q[a] \text{ est étiqueté par } i!j \\ (C_{i,j,x} - 1 \wedge I_{j,x}) \vee \neg I_{j,x} & \text{si } Q[a] \text{ est étiqueté par } j?i \end{cases}$$

La conjonction de toutes ces formules forme $\Phi_{\mathrm{p}}(\mathcal{G})$. Elle permet de représenter tous les marquages possibles à obtenir.

Propriété 5.3.6. *Soit $\mathcal{G}$ un MSG non divergent et χ' un marquage. Alors χ' est préfixe-accessible dans $\mathcal{G}$ si, et seulement si, la formule booléenne $\Phi_p(\mathcal{G})$ est satisfiable avec $C_{i,j,H} = \chi'(i,j)$ quelques soient les instances i, j.*

Démonstration. Supposons que $\Phi_{\mathrm{p}}(\mathcal{G})$ est satisfiable et $C_{i,j,H} = \chi'(i,j)$ quelles que soient les instances i, j. Considérons une affectation satisfaisant la formule $\Phi_{\mathrm{p}}(\mathcal{G})$ et telle que $C_{i,j,H} = \chi'(i,j)$ pour tout canal (i,j). Rappelons que ces variables forment un tableau représentant une exécution possible. Chaque ligne de ce tableau représente une étape de cette exécution. Soit s la suite d'états de cette exécution. Cette suite d'états représente un chemin de longueur maximale H dans $\mathcal{G}$. À chaque étape de l'exécution, c'est-à-dire à chaque ligne dans le tableau de variables, les actions étiquetées dans les états sont soit effectuées (en incrémentant ou décrémentant les compteurs de messages des canaux), soit non exécutées, mais cela implique alors que l'instance devant effectuer l'action est ou devient bloquée. L'état des compteurs de messages des canaux est ainsi mis à jour à chaque étape de l'exécution. Donc χ' est préfixe-accessible.

Supposons que χ' est préfixe-accessible dans $\mathcal{G}$. Assignons les variables booléennes de $\Phi_{\mathrm{p}}(\mathcal{G})$ telles quelles représentent un chemin de longueur maximale H dans $\mathcal{G}$ et permettent d'obtenir le préfixe χ'. Considérons un sous-ensemble d'états permettant d'obtenir le préfixe χ'. Pour chaque ligne du tableau de variables correspondant à une étape de ce sous-ensemble, nous exécutons l'action de cette étape, en incrémentant ou décrémentant la valeur des compteurs. Pour toutes les autres lignes, nous n'exécutons pas les actions et bloquons donc les instances lorsqu'elle effectue une action dans le suffixe. De cette manière, nous obtenons une assignation satisfaisant $\Phi_{\mathrm{p}}(\mathcal{G})$ et telle que $C_{i,j,H} = \chi'(i,j)$ pour chaque canal i, j. □

Ainsi, nous pouvons vérifier l'accessibilité et la couverture de marquage en ajoutant des conditions sur le nombre de messages dans les canaux pour un marquage fixé χ'. Nous considérons les formules :

$$\Phi_{\mathrm{pr}}(\mathcal{G}, \chi') = \Phi_{\mathrm{p}}(\mathcal{G}) \wedge \bigwedge_{(i,j)\in\mathcal{K}} C_{i,j,H} = \chi'(i,j)$$

$$\Phi_{\mathrm{pc}}(\mathcal{G}, \chi') = \Phi_{\mathrm{p}}(\mathcal{G}) \wedge \bigwedge_{(i,j)\in\mathcal{K}} C_{i,j,H} \geq \chi'(i,j)$$

Notons que $C_{i,j,H} = \chi'(i,j)$ et $C_{i,j,H} \geq \chi'(i,j)$ peuvent effectivement se traduire par des formules booléennes puisqu'il s'agit simplement de vérifier le codage binaire d'un nombre.

Théorème 5.3.7. *Soit $\mathcal{G}$ un MSG non-divergent et χ' un marquage. Alors*

1. *χ' est accessible par préfixe dans $\mathcal{G}$ si, et seulement si, la formule booléenne $\Phi_{pr}(\mathcal{G}, \chi')$ est satisfiable.*
2. *χ' est couvert par préfixe dans $\mathcal{G}$ si, et seulement si, la formule booléenne $\Phi_{pc}(\mathcal{G}, \chi')$ est satisfiable.*

Remarque 5.3.8. *Soit $B \in \mathbb{N}$ une taille de mémoire tampon. Le problème du canal borné pour un MSG $\mathcal{G}$* non-divergent*, une taille de mémoire tampon B, et un canal (i,j) consiste à vérifier si $W_{i,j}(\mathcal{G}) \leq B$, c'est-à-dire qu'il n'y aura jamais plus de B messages dans le canal (i,j) quelque soit l'exécution de $\mathcal{G}$. Considérons la formule*

$$\Phi_{cb}(\mathcal{G}, i, j, B) = \Phi_p(\mathcal{G}) \wedge C_{i,j,H} > B$$

Alors $W_{i,j}(\mathcal{G}) \leq B$ si, et seulement si, la formule booléenne $\Phi_{cb}(\mathcal{G}, i, j, B)$ est satisfiable.

Notons que nous pouvons effectuer une recherche dichotomique sur la taille B, afin de trouver la plus petite taille de mémoire tampon telle qu'il n'y ait aucun débordement.

Chapitre

6 Coût des réductions à SAT

Pour résoudre efficacement divers problèmes sur les MSG, nous avons formalisé ces problèmes sous la forme de formules booléennes. Les SAT-solveurs ayant fait l'objet de nombreuses années de recherche, ils sont capables de résoudre des formules booléennes de manière extrêmement efficace.

Bien que les SAT-solveurs soient très efficaces, la réduction d'un problème vers des SAT-solveurs n'est pas toujours une bonne idée. En effet, si le problème à traiter est trop éloigné d'une formulation booléenne, alors cette formulation risque d'être beaucoup plus complexe que le problème initial.

Dans la première partie de ce chapitre, nous évaluons nos réductions afin de savoir si les formulations booléennes ajoutent de la complexité par rapport au problème initial. Pour ce faire, nous introduisons la notion de *ratio*. Le principe est le suivant. Nous considérons une formule booléenne ϕ. Nous réduisons cette formule à un problème sur un MSG $\mathcal{G}$ puis nous considérons la formule booléenne ϕ' correspondant à la réduction de ce problème en une formule booléenne. Nous comparons ensuite le temps τ_ϕ et $\tau_{\phi'}$ mis pour résoudre ϕ et ϕ'. Ainsi, le ratio $\frac{\tau_{\phi'}}{\tau_\phi}$ évalue la complexité ajoutée par la double réduction pour la formule booléenne considérée. C'est une surapproximation du surcoût induit par la traduction du problème pour un MSG en une formule booléenne.

Dans la seconde partie de ce chapitre, nous évaluons le temps nécessaire pour vérifier diverses propriétés sur des MSG concrets. Nous comparons ensuite nos résultats à divers outils existants.

6.1 La notion de ratio

Soit ϕ une formule booléenne avec Q variables x_1, ..., x_Q et N clauses C_1, ..., C_N. Nous construisons un MSG $\mathcal{G}_{\text{div}}(\phi)$ avec $Q+2$ instances et $1+\sum_{k=1}^{N} d_k$ états où le degré d_k est le nombre de littéraux dans C_k. Chaque variable est vue comme une instance et nous considérons deux nouvelles instances désignées par `vrai` et `faux`. Chaque littéral de chaque clause C_k est identifié par un état et nous considérons un état supplémentaire noté $\imath$. Cet état initial est associé avec un MSC contenant un unique échange de message allant de `faux` à `vrai`. D'autre part, l'étiquette portée par les autres états l est un MSC contenant un unique échange de message allant de `vrai` vers la variable x, si $l = x$, et de x vers `faux` si $l = \text{not}(x)$. Considérons maintenant le MSG $\mathcal{G}_{\text{div}}(\phi)$ tel que nous avons

- un arc partant de chaque littéral de C_k vers chaque littéral C_{k+1} pour $1 \leq k < N$;
- un arc partant de $\imath$ vers chaque littéral C_1 et un arc allant de chaque littéral C_N vers $\imath$.

Cette construction est illustrée par un exemple sur la figure 6.1 Comme tout cycle passe par $\imath$, son graphe de communication est connexe, car il contient un arc allant de `faux` vers `vrai`.

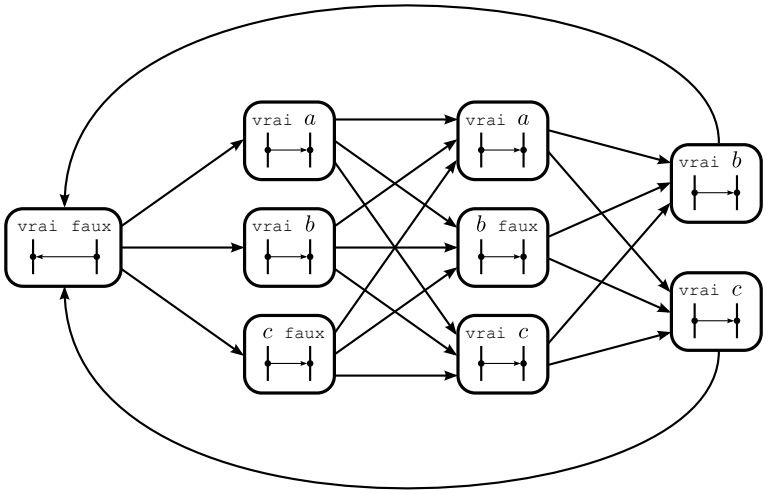


FIGURE 6.1 – Réduction de la formule $(a \vee b \vee \neg c) \wedge (a \vee \neg b \vee c) \wedge (b \vee c)$ au problème de la divergence.

De plus, il existe un chemin allant de `vrai` vers `faux` si, et seulement si, certaines variables x sont connectées à `vrai` et `faux`.

Lemme 6.1.1. *Une formule booléenne ϕ est satisfiable si, et seulement si, le canal (`faux`,`vrai`) est divergent dans $\mathcal{G}_{div}(\phi)$.*

Démonstration. Supposons d'abord que ϕ admet une affectation satisfaisante. Pour chaque clause, nous pouvons choisir un littéral satisfait. Ces choix déterminent un cycle élémentaire dans $\mathcal{G}$. Le graphe de la communication de ce cycle contient un arc allant de `faux` vers `vrai` mais aucune variable x n'admet à la fois un arc de `vrai` vers x et un arc de x vers `faux`. Ainsi, il n'y a pas de chemin de `vrai` vers `faux` : le canal $(\texttt{faux}, \texttt{vrai})$ est divergent dans $\mathcal{G}_{\text{div}}(\phi)$.

Réciproquement, supposons maintenant que le canal $(\texttt{faux}, \texttt{vrai})$ diverge dans $\mathcal{G}$. Alors il existe un cycle élémentaire dont le graphe de communication n'admet pas de chemin allant de `vrai` vers `faux`. Nous pouvons identifier un tel cycle avec le choix d'un littéral pour chaque clause. Pour chaque variable x, nous ne pouvons pas avoir d'échange de message allant de `vrai` vers x et d'échange de message allant de x vers `faux` le long de ce cycle. Ainsi, nous pouvons déduire du graphe de communication une affectation qui satisfait ϕ. □

Le théorème 5.1.2 montre que la vérification de la divergence d'un canal (i, j) d'un MSG $\mathcal{G}$ peut être réduit simplement à la satisfiabilité de $\Phi_{\text{div}}(\mathcal{G}, i, j)$. De plus, par le lemme 6.1.1, la satisfiabilité d'une formule ϕ est équivalente à la satisfiabilité de la formule $\Phi_{\text{div}}(\mathcal{G}_{\text{div}}(\phi), \texttt{faux}, \texttt{vrai})$. Ainsi, pour une formule booléenne donnée ϕ nous considérons le temps $\tau_1(\phi)$ utilisé par un SAT-solveur pour résoudre ϕ, et le temps $\tau_2(\phi)$ utilisé par le même SAT-solveur pour résoudre le problème équivalent formalisé par $\Phi_{\text{div}}(\mathcal{G}_{\text{div}}(\phi), \texttt{faux}, \texttt{vrai})$. Le ratio $\varrho_{div}(\phi) = \tau_2(\phi)/\tau_1(\phi)$ représente une estimation du coût de notre réduction pour vérifier la divergence de $\mathcal{G}_{\text{div}}(\phi)$.

Considérons deux nouvelles instances désignées par $\top$ et $\bot$. Nous remplaçons le MSC porté par l'état initial dans $\mathcal{G}_{\text{div}}(\phi)$ par un MSC contenant deux échanges de messages, un

allant de `vrai` vers $\top$, et un autre allant de `faux` vers $\bot$. On note $\mathcal{G}_{\overline{\text{gc}}}(\phi)$ le MSG résultant. Cette construction est illustrée par un exemple sur la figure 6.2. De même que pour le

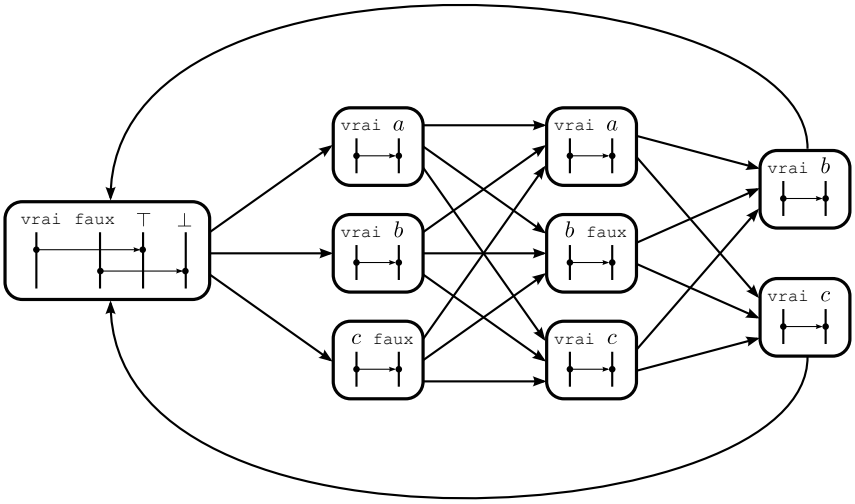


FIGURE 6.2 – Réduction de la formule $(a \vee b \vee \neg c) \wedge (a \vee \neg b \vee c) \wedge (b \vee c)$ au problème de la coopération globale.

lemme 6.1.1, nous pouvons établir le résultat suivant.

Lemme 6.1.2. *Une formule booléenne ϕ est satisfiable si, et seulement si, le MSG $\mathcal{G}_{\overline{\text{gc}}}(\phi)$ n'est pas globalement coopératif. De plus, dans ce cas le canal* (`faux`, `vrai`) *n'est pas globalement coopératif.*

Démonstration. Supposons d'abord que ϕ admet une affectation satisfaisante. Pour chaque clause, nous pouvons choisir un littéral satisfait. Ces choix déterminent un cycle élémentaire dans $\mathcal{G}_{\overline{\text{gc}}}(\phi)$. Le graphe de la communication de ce cycle contient deux composantes connexes, une première connectée à $\top$ et une seconde connectée à $\bot$. Comme aucune variable x n'admet à la fois un arc de `vrai` vers x et un arc de x vers `faux`, ces deux composantes connexes sont bien distinctes. Ainsi, le canal (`faux`, `vrai`) n'est pas globalement coopératif et donc $\mathcal{G}_{\overline{\text{gc}}}(\phi)$ n'est pas globalement coopératif.

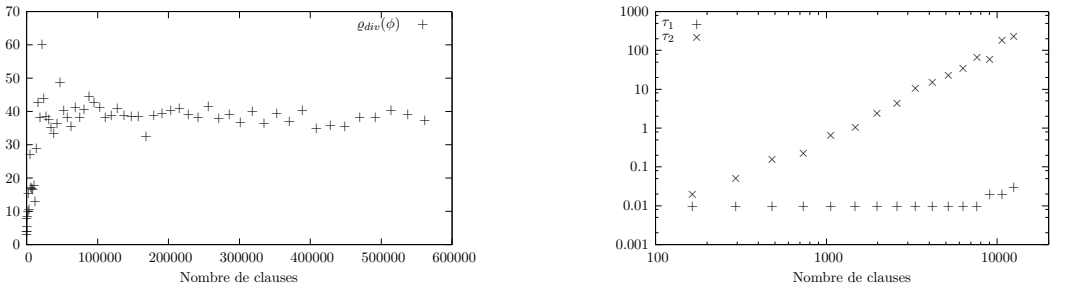
Réciproquement, supposons maintenant que le MSG $\mathcal{G}_{\overline{\text{gc}}}(\phi)$ n'est pas globalement coopératif. Alors il existe un cycle élémentaire dont le graphe de communication admet plusieurs composantes connexes. Remarquons par construction, qu'il ne peut y avoir au maximum que deux composantes connexes, une contenant `vrai` et l'autre contenant `faux`. Ainsi, un cycle élémentaire dont le graphe de communication admet plusieurs composantes connexes permet d'identifier un choix de littéral pour chaque clause. Pour chaque variable x, nous ne pouvons pas avoir d'échange de message allant de `vrai` vers x et d'échange de message allant de x vers `faux` le long de ce cycle. Ainsi, nous pouvons déduire du graphe de communication une affectation qui satisfait ϕ. □

Comme aucun état de $\mathcal{G}_{\overline{\text{gc}}}(\phi)$ n'est vide, par le théorème 5.2.4, le MSG $\mathcal{G}_{\overline{\text{gc}}}(\phi)$ n'est pas globalement coopératif si, et seulement si, la formule booléenne $\Phi_{\overline{\text{gc}}}(\mathcal{G}_{\overline{\text{gc}}}(\phi), \texttt{vrai}, \texttt{faux})$ est satisfiable. Ainsi, ϕ est satisfiable si, et seulement si, la formule $\Phi_{\overline{\text{gc}}}(\mathcal{G}_{\overline{\text{gc}}}(\phi), \texttt{vrai}, \texttt{faux})$ est satisfiable.

Pour la formule booléenne ϕ nous considérons le temps $\tau_1(\phi)$ utilisé par le SAT-solveur pour résoudre ϕ, et le temps $\tau_2(\phi)$ utilisé par le même SAT-solveur pour résoudre le problème équivalent formalisé par $\Phi_{\overline{\text{gc}}}(\mathcal{G}_{\overline{\text{gc}}}(\phi))$. Le ratio $\varrho_{gc}(\phi) = \tau_2(\phi)/\tau_1(\phi)$ représente une estimation du coût de notre réduction pour vérifier la coopération globale de $\mathcal{G}_{\overline{\text{gc}}}(\phi)$.

6.2 Coût en pratique

Nous allons considérer deux types de formules SAT afin d'évaluer les ratios ϱ_{div} et ϱ_{gc}. Le premier type de formules que nous considérons consiste à coder le problème de n-dames en formule booléenne. Le but du problème est de placer n dames sur un échiquier de $n \times n$ cases sans que les dames ne puissent se menacer mutuellement, conformément aux règles du jeu d'échec. Par conséquent, deux dames ne doivent pas partager une même rangée, colonne, ou diagonale. Simple mais non trivial, ce problème sert souvent d'exemple. Il est facile de coder ce problème en une formule booléenne Φ_n. Nous avons calculé les ratios $\varrho_{div}(\Phi_n)$ et $\varrho_{gc}(\Phi_n)$ (figure 6.3) en faisant varier le nombre n de dames.



(a) Ratio $\varrho_{div}(\phi)$ (b) Temps τ_1 et τ_2 pour la coopération globale

FIGURE 6.3 – Problème des n dames.

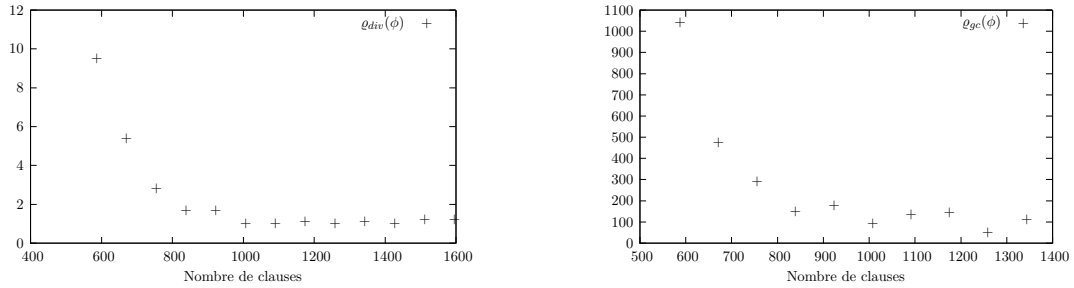
Nous pouvons observer que le ratio pour le problème de la divergence tend vers la valeur 40. En revanche ce rapport ne peut être mesuré facilement pour la coopération globale. La raison est que le temps de calcul $\tau_2(\phi)$ est tout simplement trop long quand $\tau_1(\phi)$ devient significatif, c'est-à-dire plus grand que 10ms.

Le second type de formules que nous considérons correspond à des formules générées aléatoirement. Nous générons des formules 3-SAT en choisissant un ratio de 4,2 entre le nombre de variables et le nombre de clauses. Ce ratio permet d'obtenir d'une part autant de formules satisfiables que de formules non-satisfiables et d'autre part ces formules sont souvent difficiles à résoudre [7]. Comme nous pouvons le voir sur la figure 6.4, les ratios obtenus sont bien meilleurs que pour le problème spécifique des n dames.

6.3 Comparaisons avec d'autres outils

Une autre façon naturelle d'évaluer notre approche consiste à établir une comparaison entre notre prototype et des outils existants. Nous avons considérer les trois outils suivants :

— MSCan [23], qui vérifie la coopération globale de MSG.



(a) Ratio $\varrho_{div}(\phi)$ (b) Ratio $\varrho_{gc}(\phi)$

FIGURE 6.4 – Formules 3-SAT aléatoires.

— SOFAT [6], qui vérifie simultanément plusieurs propriétés telles que la coopération globale et la divergence.

— SCStudio [21] qui vérifie la divergence de MSG.

Comme SOFAT ne permet pas de vérifier uniquement une seule propriété, il est difficile de comparer cet outil. Nous affichons cependant le temps utilisé par cet outil à titre indicatif. Nous voulions également comparer notre prototype à l'outil MESA [19], pour lequel le Pr. Leue nous a transféré les binaires, mais nous n'avons malheureusement pas pu l'exécuter. Nous avons utilisé le solveur SAT Minisat2 [2] afin de résoudre nos formules booléennes. Toutes les mesures ont été obtenues sur un ordinateur Intel® Xeon® E5620 avec 6 Go de RAM.

Nous commençons la comparaison avec des MSG aléatoires contenant n états et $n/2$ instances. Pour chaque état, nous choisissons au hasard trois états successeurs et un MSC contenant un échange de messages. La figure 6.5 montre le temps d'exécution moyen pour :

— MSCan pour vérifier la coopération globale (τ_{MSCan}),

— SOFAT pour vérifier diverses propriétés incluant la coopération globale et la divergence (τ_{SOFAT}),

— SCStudio pour vérifier la divergence ($\tau_{ScStudio}$),

— Minisat2 pour vérifier la divergence (τ_{DIV}) à l'aide de la formule $\Phi_{div}(\mathcal{G})$ du théorème 5.1.2,

— Minisat2 pour vérifier la coopération globale (τ_{GC}) à l'aide de la formule $\Phi_{\overline{\text{gc}}}(\mathcal{G})$ du théorème 5.2.4.

Nous considérons ensuite les mêmes outils pour le protocole des fenêtres coulissantes de la figure 4.1 avec diverses tailles de fenêtre (figure 6.6). Ce protocole est globalement coopératif et non-divergent.

Pour terminer, nous considérons les MSG générés à partir de la réduction du problème des n dames vers la coopération globale (figure 6.7).

Nous pouvons constater que les résultats obtenus sont très encourageants. Pour les trois types d'instances, notre méthode permet de résoudre en moins d'une seconde des instances que les autres outils n'arrivent pas à résoudre en un temps raisonnable. L'utilisation d'un solveur SAT pour résoudre la divergence ou la coopération globale permet également de

Nombre d'états	τ_{DIV}	τ_{GC}	τ_{MSCan}	τ_{SOFAT}	$\tau_{ScStudio}$
7	< 0.01	< 0.01	0.5	1	< 1
8	< 0.01	< 0.01	0.6	4	< 1
9	< 0.01	< 0.01	0.65	3	< 1
10	< 0.01	< 0.01	0.8	14	< 1
11	< 0.01	< 0.01	0.94	78	< 1
12	< 0.01	< 0.01	1.1	191	< 1
13	< 0.01	< 0.01	1.3	841	< 1
14	< 0.01	< 0.01	2.5	2700	1
15	< 0.01	< 0.01	3.5		2
16	< 0.01	< 0.01	14		1
17	< 0.01	< 0.01	21		9
18	< 0.01	< 0.01	58		9
19	< 0.01	< 0.01	210		40
20	< 0.01	< 0.01	440		450
21	< 0.01	< 0.01	550		80
22	< 0.01	< 0.01	1800		3000
23	< 0.01	< 0.01	2700		
100	< 0.01	0.225			
1 000	2	140			

FIGURE 6.5 – Temps moyen en sec. pour des MSG aléatoires.

Taille de la fenêtre	τ_{DIV}	τ_{GC}	$\tau_{\text{SC Studio}}$	τ_{SOFAT}	τ_{MSCan}
70	< 0.1	< 0.1	< 0.1	37	50
80	< 0.1	< 0.1	< 0.1	60	86
90	< 0.1	< 0.1	< 0.1	98	140
100	< 0.1	< 0.1	< 0.1	150	210
1 000	< 0.1	< 0.1	5		
2 000	0.1	0.1	20		
3 000	0.2	0.2	45		
10 000	0.5	0.7			
20 000	1.2	1.5			
40 000	2.6	3.8			
80 000	5.9	10			
160 000	14	30			

FIGURE 6.6 – Temps en sec. pour le protocole de la fenêtre coulissante.

n	Clauses	États	τ_{GC}	τ_{MSCan}	τ_{SOFAT}
2	8	25	<0.1	5.17	190
3	31	97	<0.1	94 280.67	$>$ 100 000
4	80	249	<0.1		
8			0.3		
12			4.3		
16			35		
20			230		

FIGURE 6.7 – Coopération globale codant le problème des n dames.

résoudre de très grosses instances. Par exemple, pour les instances concrètes du protocole simplifié des fenêtres coulissantes, nous somme capable de vérifier la divergence et la coopération globale pour des tailles de fenêtre dépassant facilement 100000 alors que les autres outils sont limités à des tailles de quelques milliers.

Deuxième partie

Réseaux de Petri avec états

Chapitre

7 Introduction

Considérons un ensemble de réactions qui ont lieu au sein d'une collection de particules telles que chaque réaction consomme un multi-ensemble de particules disponibles et produit une combinaison linéaire d'autres types de particules. Considérons de plus un état de contrôle qui détermine si une règle peut se produire ou non et tel que l'exécution de cette règle conduit à un nouvel état de contrôle, éventuellement identique. Ce modèle correspond au formalisme des PNS que nous avons introduits dans les préliminaires (section 2.5).

Le modèle des MSG, étudié dans la première partie, peut être considéré comme un cas particulier de PNS où les réactions autorisées sont uniquement l'envoi et la réception de messages entre des instances. Chaque séquence de règles peut être décrite par un ordre partiel d'événements appelé un MSC (définition 3.1.1). Chaque MSC correspond à plusieurs séquences de règles élémentaires qui sont équivalentes quitte à réordonner des événements indépendants. De même, chaque séquence de MSC est équivalente à plusieurs séquences de MSC. Ainsi, les états de contrôle sont utilisés pour se concentrer sur des entrelacements particuliers d'événements. Il n'existe à ce jour aucun moyen de considérer l'exécution d'un PNS (ou d'un VASS) comme un ordre partiel d'événements. Par conséquent, il n'existe aucun moyen d'appliquer les techniques ou les outils des réseaux de Petri à l'analyse de MSG. Dans ce chapitre, nous étudions une sémantique d'ordres partiels pour les PNS de telle sorte que le cadre des MSG puisse effectivement être considéré comme un cas particulier de PNS et que les techniques des réseaux de Petri puissent s'appliquer aux MSG. Contrairement aux MSG généralement étudiés, nous obtenons un cadre formel où les compteurs et les contraintes temporelles (avec un temps discret) peuvent être facilement modélisés et analysés.

Proposée par C.A. Petri dans le cadre restreint de systèmes de condition/événement [88], la sémantique des processus d'un réseau de Petri définit un graphe d'occurrence étiqueté comme un ensemble partiellement ordonné d'événements avec une condition de non-branchement [20, 42, 54, 91, 103]. Contrairement à l'autre sémantique d'ordres partiels classique, basée sur des séquences d'ensemble de règles [55, 66, 103], un processus enregistre toutes les dépendances causales entre les événements qui se produisent le long d'une exécution. Nous présentons dans ce chapitre une sémantique d'ordres partiels pour les PNS qui étend la sémantique habituelle des processus des réseaux de Petri. L'approche est simple et naturelle. D'abord, nous considérons l'ensemble des séquences de règles franchissables d'un PNS, ensuite nous définissons les processus qui représentent une séquence donnée. Ainsi, chaque processus décrit des dépendances causales entre les événements qui ne sont plus totalement ordonnés. Cela signifie que deux règles qui apparaissent l'une après l'autre dans une séquence de règles peuvent se produire simultanément au sein d'un processus. Cette situation est classique dans la modélisation de systèmes asynchrones. En particulier, cela est similaire à la façon dont les MSC d'un MSG sont obtenus (section 3.3). En particulier, les états de contrôle représentent des étapes abstraites de calculs utilisées pour spécifier des ensembles de séquences de règles : ils n'apparaissent pas formellement dans la sémantique

des processus. De cette façon, les MSG sont inclus dans le cadre des PNS. Une caractéristique spécifique de la sémantique des processus est qu'une séquence de règles peut donner en général plusieurs processus non isomorphes en fonction de l'ordre de consommation de particules identiques. Dans cette partie, nous allons exposer quelques autres faits qui montrent clairement que le modèle de PNS est plus général et plus difficile à manipuler que celui des MSG et que celui des réseaux de Petri.

Nous montrerons dans ce chapitre que l'inclusion (ou l'égalité) de deux langages de processus donnés par deux PNS est indécidable. La raison essentielle est que l'égalité et l'inclusion de langages rationnels de traces de Mazurkiewicz [31] sont indécidables, car le problème de l'universalité est indécidable [95]. De plus, les langages rationnels de traces de Mazurkiewicz peuvent être représentés par des MSG [59] et les MSG sont inclus dans les PNS. Ce résultat illustre l'écart entre les réseaux de Petri et les PNS sous la sémantique des processus. En effet, ces deux problèmes sont décidables pour les réseaux de Petri, au moyen d'une réduction simple au problème de couverture [43]. Cela montre également que l'analyse des exécutions partiellement ordonnées d'un PNS ne se réduit pas à la vérification d'un réseau de Petri en général, en dépit de la simulation bien connue d'un PNS par un réseau de Petri (section 2.6).

Les problèmes de synthèse ont été étudiés pour différents modèles de concurrence : automate asynchrone [31, 40, 104], réseaux de Petri [35, 41, 61, 83], machines communicantes à états finis [11, 59], etc. Elles consistent principalement à caractériser les comportements formels correspondant à une catégorie de dispositifs concurrents et de construire, s'il existe, un tel dispositif à partir de ses caractéristiques comportementales. Nous étudions aussi dans la section 7.4 un problème de synthèse naturel : étant donné un PNS, nous nous demandons si ses processus peuvent être générés par un réseau de Petri. Nous montrons que ce problème est indécidable (même pour les systèmes bornés) au moyen d'une réduction au problème de l'inclusion. Cependant, nous présentons dans la suite de cette partie plusieurs nouvelles techniques qui permettent de vérifier d'autres propriétés d'un PNS sous la sémantique des processus.

Le problème de model-checking des MSG pour la logique monadique du second ordre (MSO) a été étudié pour la première fois dans [76]. Contrairement aux travaux précédents [11], les formules sont interprétées sur des scénarios partiellement ordonnés acceptés par les MSG. Ce problème est décidable pour la classe des MSG compositionnels sûrs [77, 50], c'est-à-dire les MSG considérés dans cette thèse (section 3.4). Chaque MSG peut être considéré comme un PNS borné. Toutefois, un MSG peut décrire un ensemble infini de marquages, car un MSG peut être divergent. Nous présentons dans le chapitre 8 une technique permettant de vérifier que tous les processus d'un PNS borné donné satisfont une formule MSO donnée.

Nous avons étudié dans la première partie de cette thèse le problème de la détection de la divergence d'un canal, à savoir, de décider si le nombre de messages en attente au cours des exécutions est non borné [11, 13, 18, 59]. Ce problème est NP-complet pour les MSG [11, Théorème 7]. Un problème équivalent, dans le cadre plus général des PNS est le problème du caractère borné des préfixes. Cela consiste à vérifier si l'ensemble des marquages atteints par les *préfixes* des processus est fini. Nous présentons dans le chapitre 9 une technique

pour résoudre ce problème au moyen d'une réduction au problème de décider si un réseau de Petri est borné. Soulignons que notre construction diffère de la simulation habituelle d'un PNS (ou d'un VASS) par un réseau de Petri, car *cette dernière ne conserve pas le caractère borné des préfixes*. Le problème de la borne des préfixes est donc équivalent au problème de la borne pour les réseaux de Petri et nécessite un espace exponentiel [43]. Ce résultat présente un écart de complexité intéressant entre les MSG et les PNS. Il montre que les algorithmes vérifiant les propriétés des MSG doivent être améliorés dans le cadre plus expressif des PNS. D'autres problèmes de décision de base sur les marquages atteints par les préfixes sont bien sûr intéressants. Nous montrons en particulier que l'accessibilité et la couverture d'un marquage donné par les préfixes d'un PNS peuvent être résolues en utilisant la même approche.

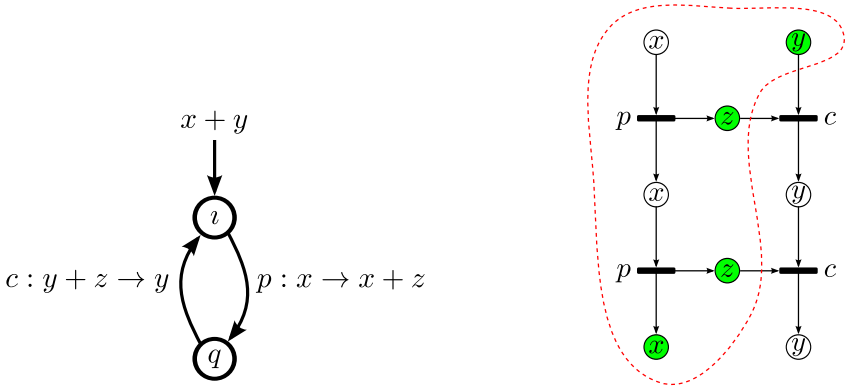
Dans le reste de ce chapitre, nous commençons par introduire formellement la sémantique adoptée pour les PNS. Nous montrons comment sont construits les processus d'un PNS sous cette sémantique. Ensuite, nous montrons comment les PNS sous la sémantique des processus permettent de généraliser les MSG. Pour terminer, en considérant les PNS comme un moyen de spécifier des comportements parallèles sous la forme de processus, nous abordons le problème consistant à construire un réseau de Petri équivalent à un PNS.

Par souci de simplicité, pour toute application $\lambda : A \to B$ entre deux ensembles finis A et B, nous noterons également par λ l'application naturelle $\lambda : A^* \to B^*$ des mots sur A vers les mots sur B et l'application $\lambda : \mathbb{N}^A \to \mathbb{N}^B$ d'un multi-ensemble A vers un multi-ensemble B telle que $\lambda(\mu) = \sum_{a \in A} \mu(a) \cdot \lambda(a)$ pour chaque multi-ensemble $\mu \in \mathbb{N}^A$. De plus, nous identifierons un ensemble S avec un multi-ensemble μ_S pour lequel $\mu_S(x) = 1$ si $x \in S$ et $\mu_S(x) = 0$ sinon.

7.1 La sémantique des processus

Dans cette partie, nous nous intéressons à une sémantique de PNS basée sur les réseaux causaux qui est une généralisation directe de la sémantique des processus des réseaux de Petri [20, 42, 53, 54, 91, 103]. Un processus d'un réseau de Petri $\mathcal{N}$ est un réseau causal $\mathcal{K}$ dans lequel chaque condition de $\mathcal{K}$ est étiquetée par une place de $\mathcal{N}$ et chaque événement de $\mathcal{K}$ est étiqueté par une transition de $\mathcal{N}$. La sémantique des processus des réseaux de Petri caractérise les réseaux causaux étiquetés qui décrivent l'exécution d'un réseau de Petri donné. Nous avons déjà observé que chaque transition d'un réseau de Petri peut être considérée comme une règle. Pour cette raison, nous adoptons une représentation graphique des règles similaires aux transitions des réseaux de Petri, comme le montre la figure 7.2(a). Étant donné un multi-ensemble initial de places, chaque séquence de règles tirable peut être représentée par un réseau causal, appelé processus, qui en quelque sorte recolle les représentations de chaque règle. Par exemple, le réseau causal étiqueté $\mathcal{K}$ de la figure 7.1(b) représente un processus du réseau de Petri $\mathcal{N}$ de la figure 2.5(b).

La définition qui suit explique comment les processus sont obtenus à partir d'une séquence de règles donnée. Ensuite, les processus d'un PNS seront définis comme les processus de ses séquences de règles tirables (définition 7.1.2).



(a) Un PNS avec deux états de contrôle (b) Un processus et un préfixe

FIGURE 7.1 – Exemple du producteur consommateur.

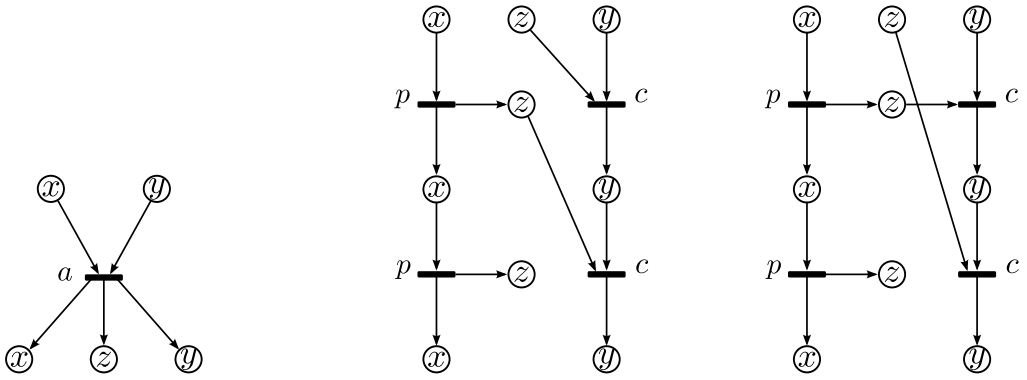
Définition 7.1.1. *Un* processus *d'une séquence de règles $s = r_1...r_n \in R^\star$ à partir d'un marquage $\mu \in \mathbb{N}^P$ se compose d'un réseau causal $\mathcal{K} = (B, E, F, \mu_{min})$ avec n événements $e_1, ..., e_n$ et d'un étiquetage $\pi : B \cup E \to P \cup N$ tels que les conditions suivantes sont satisfaites :*

1. *$\pi(b) \in P$ pour tout $b \in B$, $\pi(e) \in N$ pour tout $e \in E$, et $\pi(\mu_{min}) = \mu$;*
2. *$r_i = (\pi(e_i), \pi({}^\bullet e_i), \pi(e_i^\bullet))$ pour tout $i \in [1, n]$;*
3. *$e_i F^+ e_j$ implique $i < j$ pour chaque $i, j \in [1, n]$.*

On note par $[\![s]\!]_\mu$ la classe de tous les processus de s à partir du marquage μ.

Dans cette définition, π désigne l'étiquetage de $\mathcal{K}$ et son extension naturelle aux multi-ensembles. La première condition affirme que le marquage initial du réseau causal décrit le marquage μ. De plus, chaque condition est associée à une place et chaque événement correspond à un nom de la règle. La deuxième condition exige une correspondance entre les règles et les événements. Enfin, la dernière propriété assure que l'ordre total des règles dans s correspond à une extension de l'ordre partiel des événements dans $\mathcal{K}$. Par conséquent, tout sous-ensemble d'événements $\{e_1, ..., e_k\}$ est fermé inférieurement. C'est donc une *configuration* (page 12). De plus, le réseau causal préfixe $\mathcal{K}'$ correspondant à la configuration $\{e_1, ..., e_{n-1}\}$ (voir la définition du préfixe d'un réseau causal à la page 13) est un processus de la séquence de règles $r_1...r_{n-1}$ à partir du même marquage μ. Par conséquent, la classe des processus d'une séquence de règles pourrait aussi être définie inductivement sur sa longueur, comme nous le verrons dans la propriété 7.2.1. En outre, nous verrons que la classe des processus d'une séquence de règles est vide si, et seulement si, la séquence de règles n'est pas tirable à partir de μ_{min}.

Soit H une configuration d'un processus $\mathcal{K} = (B, E, F, \mu_{min}, \pi)$ pour une séquence de règles s à partir de μ. Soit $B_{\max}$ l'ensemble des conditions maximales du préfixe $\mathcal{K}_H$ par rapport à F^*. Alors, le multi-ensemble de places $\pi(B_{\max})$ est appelé *marquage atteint par* $\mathcal{K}_H$ et on dit que $\mathcal{K}_H$ *conduit au marquage* $\pi(B_{\max})$. Soit s_H une extension linéaire des événements présents dans H. Alors, il est clair que la séquence de règles $\pi(s_H)$ est tirable à



(a) Règle $a : x+y \to x+y+z$ (b) Deux processus partant de $x + y + z$

FIGURE 7.2 – Illustration des règles et des processus.

partir de μ et conduit au marquage $\pi(B_{\max})$. De plus, $\mathcal{K}_H$ est un processus de la séquence de règles $\pi(s_H)$ partant de μ.

Formellement, tout réseau causal étiqueté isomorphe à un processus de s est aussi un processus de s. En particulier, la classe des processus de la séquence de règles vide partant d'un marquage μ recueille tous les réseaux causaux étiquetés avec aucun événement et tels que l'ensemble de ses places représente le multi-ensemble μ. Par ailleurs une séquence de règles peut donner lieu à de multiples réseaux causaux (non isomorphes) en fonction de la consommation des jetons de chaque événement et du marquage initial. Par exemple, la séquence de règles $(p : x \to x + z) \cdot (c : y + z \to y) \cdot (p : x \to x + z) \cdot (c : y + z \to y)$ du PNS de la figure 7.1(a) correspond au réseau causal de la figure 7.1(b). Cependant, s'il y a $x + y + z$ jetons au départ, alors cette séquence de règles correspond aux deux réseaux causaux étiquetés de la figure 7.2(b) parmi d'autres.

Définition 7.1.2. *Soit $\mathcal{S}$ un PNS avec un marquage initial μ_{in}. Un* processus de $\mathcal{S}$ *est un processus d'une séquence de règles de $\mathcal{S}$ à partir de μ_{in}. Nous notons $[\![\mathcal{S}]\!]$ la classe de tous les processus de $\mathcal{S}$.*

Ainsi $[\![\mathcal{S}]\!] = \bigcup_{s \in CS(\mathcal{S})[\![s]\!]_{\mu_{in}}}$. Il est facile de vérifier que les processus d'un PNS munis d'un seul état sont précisément les processus du réseau de Petri correspondant selon la sémantique des processus habituelle [20, 53, 103]. En outre, tout préfixe d'un processus de $\mathcal{S}$ est un processus d'une séquence de règles. Par conséquent, la classe des processus des réseaux de Petri est close par préfixe. Cependant *l'ensemble des processus d'un PNS n'est pas clos par préfixe en général*, comme le montre l'exemple suivant.

Exemple 7.1.3. *Considérons le PNS de la figure 7.1(a) avec un marquage initial $x + y$ et son processus décrit sur la figure 7.1(b). Clairement, le préfixe de ce processus encerclé par la ligne en pointillés sur la figure 7.1(b) n'est pas un processus du PNS car p.p n'est pas une séquence de règles tirable dans $\mathcal{S}$.*

Rappelons qu'un PNS est borné si l'ensemble de ses marquages accessibles est fini (page 14). Un PNS est dit *borné par préfixe* si l'ensemble des marquages accessibles par

des préfixes est fini. Clairement, les PNS bornés par préfixe sont bornés. Cependant, la propriété inverse n'est pas vraie en général. Continuons l'exemple 7.1.3 : chaque processus du PNS de la figure 7.1(a) conduit à un marquage avec au plus 3 jetons alors que les préfixes de ces processus conduisent à une infinité de marquages distincts (voir dans la figure 7.1(b) un préfixe du processus qui conduit à un marquage avec 4 jetons). Cependant, soulignons que chaque réseau de Petri borné est borné par préfixe, car sa classe de processus est close par préfixe.

Notons que la simulation d'un PNS par un réseau de Petri décrite dans la sous-section 2.7 n'est pas fidèle à l'ordre partiel du point de vue que nous adoptons ici. Considérons à nouveau la figure 7.3. Les processus du PNS $\mathcal{S}$ (avec trois places) sur le côté gauche diffèrent des processus du réseau de Petri $\mathcal{N}$ (avec cinq places) sur le côté droit. Dans le chapitre 9, nous introduirons une simulation d'un PNS par un autre PNS qui nous permettra d'analyser l'ensemble des marquages accessibles par les préfixes des processus d'un PNS donné.

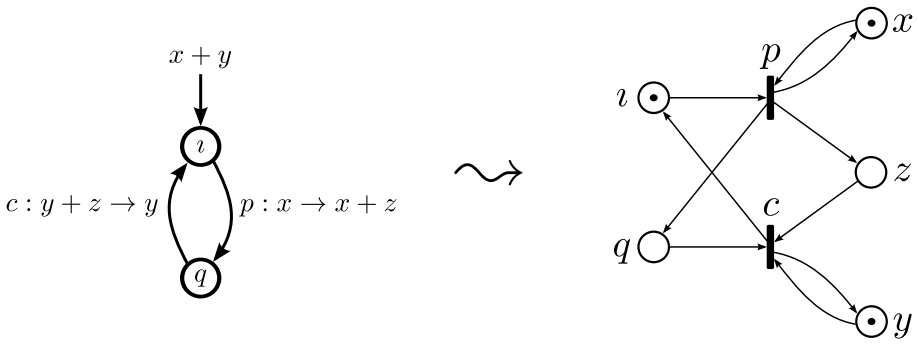


FIGURE 7.3 – Simulation d'un PNS par un réseau de Petri.

7.2 Construction inductive des processus

Pour chaque séquence de règles $u = r_1...r_n \in R^\star$, le coût de u correspond au vecteur $\text{cost}(u) \in \mathbb{Z}^P$ telle que $\text{cost}(u)(p) = \sum_{i=1}^{n} {}^\bullet r_i(p) - r_i{}^\bullet(p)$ pour chaque $p \in P$. En particulier, le coût de la séquence de règles vide est le vecteur nul. Si une séquence de règles u est tirable à partir d'un marquage μ, alors le marquage atteint par u depuis μ est $\mu - \text{cost}(u)$. Soit $\mathcal{K}$ un processus d'une séquence de règles tirable u à partir de μ. Alors $\mu - \text{cost}(u)$ est le marquage atteint par $\mathcal{K}$ depuis μ. En outre, le marquage $\mu - \text{cost}(u)$ correspond à l'ensemble des conditions de $\mathcal{K}$ qui ne sont pas des places d'entrées d'un événement dans $\mathcal{K}$ (ce qui signifie intuitivement qu'ils sont encore disponibles), c'est-à-dire aux places maximales de $\mathcal{K}$.

Pour chaque règle r telle que ${}^\bullet r \leq \mu - \text{cost}(u)$, nous désignons par $\mathcal{K} \cdot r$ la classe des réseaux causaux étiquetés obtenus en ajoutant à $\mathcal{K}$ un événement qui décrit une occurrence de la règle r qui consomme ${}^\bullet r$ conditions disponibles dans $\mathcal{K}$.

Propriété 7.2.1. *Soit $\mu \in \mathbb{N}^P$. La classe des processus d'une séquence de règles $u \in R^\star$ satisfait les trois propriétés suivantes :*

— *Si u est vide, c'est-à-dire $u = \varepsilon$, alors chaque processus de $[\![\varepsilon]\!]_\mu$ se compose de $\sum_{p \in P} \mu(p)$ conditions et d'aucun événement.*

— *Pour toutes règles $r \in R$, l'ensemble des processus $[\![u.r]\!]_\mu$ est vide si $[\![u]\!]_\mu$ est vide ou* ${}^\bullet r \gneq \mu - cost(u)$.

— *Pour toutes règles $r \in R$, si l'ensemble des processus $[\![u]\!]_\mu$ n'est pas vide et* ${}^\bullet r \leq \mu - cost(u)$ *alors $[\![u.r]\!]_\mu$ recueille tous les processus de $\mathcal{K} \cdot r$ pour tous les processus $\mathcal{K} \in [\![u]\!]_\mu$.*

Démonstration. Par la définition 7.1.1, un processus de la séquence de règles vide à partir d'un marquage μ se compose d'un ensemble des places qui représente μ. Considérons une séquence de règles u et une règle r. Supposons que $[\![u.r]\!]_\mu$ ne soit pas vide. Soit $\mathcal{K}$ un réseau causal étiqueté de $[\![u.r]\!]_\mu$. Nous avons déjà remarqué que certains préfixes $\mathcal{K}'$ de $\mathcal{K}$ sont des processus de u depuis μ. Par conséquent, $[\![u]\!]_\mu$ n'est pas vide. De plus, $\mathcal{K}$ appartient à $\mathcal{K}' \cdot r$. Comme $u.r$ est une séquence de règles tirable, ${}^\bullet r$ est plus petit que les marquages atteints par u depuis μ, c'est-à-dire ${}^\bullet r \leq \mu - \mathrm{cost}(u)$. Ainsi, par contraposition, si $[\![u]\!]_\mu$ est vide ou ${}^\bullet r > \mu - \mathrm{cost}(u)$, alors $[\![u.r]\!]_\mu$ est vide. En revanche, si $[\![u]\!]_\mu$ n'est pas vide et ${}^\bullet r \leq \mu - \mathrm{cost}(u)$, alors tout réseau causal étiqueté de $\mathcal{K}' \cdot r$ avec $\mathcal{K}' \in [\![u]\!]_\mu$ est clairement un processus de $u.r$ depuis μ. De plus, tout processus de $[\![u.r]\!]_\mu$ peut-être obtenu de cette manière. □

Étant donné deux multi-ensembles de places $\mu_1, \mu_2 \in \mathbb{N}^P$, le maximum $\max(\mu_1, \mu_2)$ est égal au nombre maximal de jetons dans chaque place : $\max(\mu_1, \mu_2)(p) = \max(\mu_1(p), \mu_2(p))$ pour chaque $p \in P$. Nous allons utiliser par la suite la *fonction de prérequis* $\mathrm{req} : R^\star \to \mathbb{N}^P$.

Définition 7.2.2. *Le* prérequis *d'une séquence de règles $u \in R^\star$ est le multi-ensemble de places défini inductivement de la manière suivante :*

— $req(\varepsilon) = 0$,

— $req(u.a) = \max(req(u), {}^\bullet a + cost(u))$ *pour tout $u \in R^\star$ et tout $a \in R$.*

L'observation suivante montre que le prérequis d'une séquence de règles u est le marquage minimal μ tel que u est tirable à partir de μ c'est-à-dire que $[\![u]\!]_\mu \neq \emptyset$.

Propriété 7.2.3. *Soit $u \in R^\star$ et $\mu \in \mathbb{N}^P$. Alors $[\![u]\!]_\mu \neq \emptyset$ si, et seulement si, $\mu \geq req(u)$.*

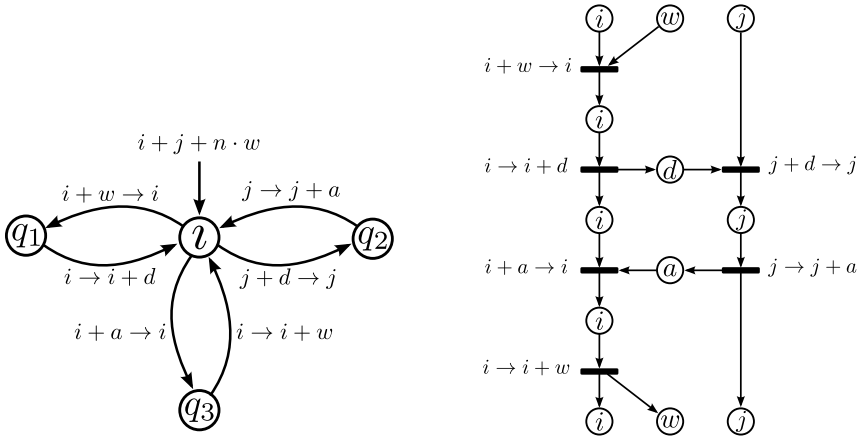
Démonstration. Nous procédons par induction sur la longueur de u. Si $u = \varepsilon$ alors $\mathrm{req}(u) = 0$ d'où $\mu \geq \mathrm{req}(u)$. De plus, $[\![\varepsilon]\!]_\mu$ contient chaque réseau causal avec aucun événement et avec un ensemble de conditions représentant le marquage μ. Étape d'induction : soit $u \in R^\star$ et $a \in R$. Supposons pour commencer que $[\![u.a]\!]_\mu \neq \emptyset$. Par la propriété 7.2.1, nous avons $\mu \geq {}^\bullet a + \mathrm{cost}(u)$. D'un autre coté, nous avons $[\![u]\!]_\mu \neq \emptyset$ d'où $\mu \geq \mathrm{req}(u)$ par hypothèse d'induction. Ainsi, $\mu \geq \max(\mathrm{req}(u), {}^\bullet a + \mathrm{cost}(u)) = \mathrm{req}(u.a)$. Supposons maintenant que $[\![u.a]\!]_\mu = \emptyset$. Nous distinguons deux cas.

1. $[\![u]\!]_\mu = \emptyset$. Alors, par hypothèse d'induction, nous avons $\mu < \mathrm{req}(u) \leq \mathrm{req}(u.a)$.
2. $[\![u]\!]_\mu \neq \emptyset$. Alors, par la propriété 7.2.1, nous avons $\mu < {}^\bullet a + \mathrm{cost}(u) \leq \max(\mathrm{req}(u), {}^\bullet a + \mathrm{cost}(u)) = \mathrm{req}(u.a)$.

Ainsi $[\![u.a]\!]_\mu \neq \emptyset$ si, et seulement si, $\mu \geq \mathrm{req}(u.a)$. □

7.3 Des MSG aux PNS

Considérons un système distribué composé d'un ensemble $\mathcal{I}$ d'instances et d'un ensemble K de canaux de communications entre les instances. Le comportement d'un tel système peut être spécifié par un PNS sur l'ensemble de places $P = \mathcal{I} \cup K$ telles que les envois de messages à partir de l'instance i vers l'instance j dans le canal $k_{i,j}$ de i vers j est codé par une règle $i!j : i \to i + k_{i,j}$ et la réception d'un tel message est codé par une règle $j?i : j + k_{i,j} \to j$. Nous exigeons que le marquage initial (et chaque marquage accessible) contienne un seul jeton dans chaque place $i \in \mathcal{I}$, de sorte que tous les événements d'un processus localisés sur une instance donnée sont totalement ordonnés. Un tel PNS peut être considéré comme un MSG dont les comportements ne sont pas nécessairement FIFO.



(a) Modélisation avec un PNS

(b) Un processus avec $n = 1$

FIGURE 7.4 – Protocole simplifié des fenêtres coulissantes.

Exemple 7.3.1. *Le PNS de la figure 7.4(a) décrit une version simplifiée du protocole des fenêtres coulissantes utilisé pour transmettre des données d'un serveur i à un client j. Le nombre maximal d'acquittements manquants est formalisé par n jetons initiaux sur la place w (la fenêtre ou « window »). Le comportement du système se décompose en trois étapes.*

1. *Le serveur envoie un nouveau message représenté par un jeton d si des jetons w sont disponibles : il consomme un jeton w $(i + w \to i)$, et envoie la donnée $(i \to i + d)$.*
2. *Le client reçoit un message et retourne un acquittement représenté par un jeton a : il consomme la donnée $(j + d \to j)$, et produit un acquittement $(j \to j + a)$.*
3. *Le serveur reçoit un acquittement et incrémente la taille de la fenêtre : l'acquittement est consommé $(i + a \to i)$, et un nouveau jeton w est créé $(i \to i + w)$.*

Un processus typique de ce système avec $n = 1$ est représenté dans la figure 7.4(b)

Comme les compteurs sont proscrits dans les MSG étudiés dans la première partie, tout MSG équivalent au PNS de l'exemple 7.3.1 a besoin de n états distincts pour représenter l'état courant de la fenêtre. Un tel MSG est donc exponentiellement plus grand que ce PNS du fait que n est codé en binaire : si ce protocole commence avec une taille de fenêtre initiale

de $n = 2^k \cdot w$, alors tout MSG décrivant la même classe de processus a besoin de 2^k états distincts.

Les PNS apparaissent ainsi comme une généralisation des MSG qui permet d'obtenir des spécifications plus concises. Voici comment nous pouvons définir formellement le plongement des MSG dans les PNS.

Définition 7.3.2. *Un PNS est équivalent à un MSG s'il existe un ensemble d'instances $\mathcal{I}$ tel que :*

- *Chaque place correspond à un canal entre deux instances ou à une instance : $P = \mathcal{I} \times \mathcal{I} \setminus \{(i,i)|i \in \mathcal{I}\} \uplus \mathcal{I}$.*
- *les seules règles existantes sont les envois et les réceptions de messages :*
  - $i!j : i \to i + (i,j)$
  - $i?j : i + (j,i) \to i$
- $\mu_{in}(i) = 1$ *pour chaque* $i \in \mathcal{I}$.

Remarquons que les règles des PNS sont présentes sur les arcs tandis que celles des MSG sont présentes dans les états. Toutefois, cela ne change rien essentiellement à l'expressivité du modèle. De plus, dans cette définition, nous omettons le fait que les MSG considérés dans la première partie attribuent un marquage fixe à chaque état.

Dans cette seconde partie, nous considérons à plusieurs reprises une sous-classe de PNS qui généralise les MSG et que nous appelons des MSG étendus.

Définition 7.3.3. *Un PNS est un* MSG étendu *s'il existe un ensemble d'instances $\mathcal{I}$ tel que :*

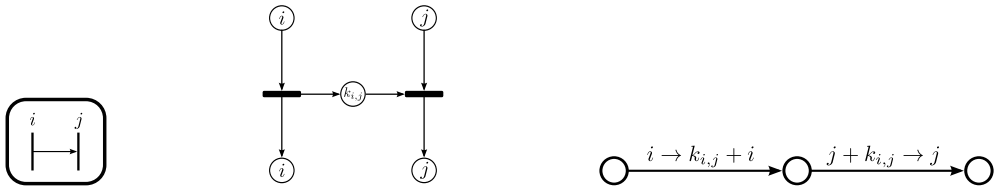
- *Il existe un ensemble de places correspondant à des canaux entre deux instances ou à une instance : $P \supseteq \mathcal{I} \times \mathcal{I} \setminus \{(i,i)|i \in \mathcal{I}\} \uplus \mathcal{I}$.*
- *Toutes les règles utilisant des places canaux sont des envois ou des réceptions de messages du type $i!j : i \to i + (i,j)$ ou $i?j : i + (j,i) \to i$. Les autres règles n'affectent pas les canaux.*
- $\mu_{in}(i) = 1$ *quelque soit l'instance* $i \in \mathcal{I}$.
- *Si une règle $r = (\lambda, \alpha, \beta)$ consomme une instance i, alors cette même règle restitue une instance i et inversement : $i \in \alpha \Leftrightarrow i \in \beta$ pour chaque $i \in \mathcal{I}$. Il y aura donc toujours un et un seul jeton dans chaque place instance.*
- *Si une règle r utilise une instance i et une place a ne correspondant pas à un canal, alors la place a est toujours utilisée avec l'instance i et aucune autre instance ; autrement dit,*

  pour chaque place $a \in P \setminus (\mathcal{I} \times \mathcal{I})$, et chaque instance $i \in \mathcal{I}$:

  - $(\exists r | a \in {}^\bullet r \cup r^\bullet \wedge i \in {}^\bullet r) \Rightarrow (\forall r : a \in {}^\bullet r \cup r^\bullet \Rightarrow i \in {}^\bullet r)$
  - $(\exists r | a \in {}^\bullet r \cup r^\bullet \wedge i \in {}^\bullet r) \Rightarrow (\forall r, \forall j \in \mathcal{I} : (a \in {}^\bullet r \cup r^\bullet \wedge j \in {}^\bullet r) \Rightarrow i = j)$

  Intuitivement, cette dernière condition garantit que les places qui ne sont ni des instances ni des canaux sont ou bien localisées sur une instance précise ou indépendante de toute instance.

Afin de rendre la représentation des MSG étendus plus intuitive, nous ajoutons la possibilité de dessiner des MSC à l'intérieur des nœuds du graphe. Nous supposons cependant que les modèles obtenus sont bien équivalents à un MSG étendus. Un échange de message entre deux instances peut ainsi être représenté par deux règles comme le montre la figure 7.5.



(a) MSC (b) Processus (c) Règles

FIGURE 7.5 – La représentation sous forme de MSC peut être modélisée avec des règles.

Nous ajoutons également la possibilité d'ajouter directement des affectations et des gardes sous forme d'égalité ou d'inégalité à respecter sur des variables entières positives fixées. Nous imposons cependant que la valeur maximale de chaque variable soit définie. Il est alors simple de modéliser des affectations ou des conditions d'inégalité à respecter. Pour cela, nous devons utiliser deux places pour modéliser une variable. Par exemple, pour modéliser une variable v_a, nous pouvons utiliser les places a et $\bar{a}$. Le nombre de jetons dans a sera égal à la valeur de v_a. Le nombre de jetons dans $\bar{a}$, noté $v_{\bar{a}}$, sera égal au complémentaire de v_a pour atteindre la valeur maximale a_{max}. Autrement dit, à tout moment $v_a + v_{\bar{a}} = a_{max}$. Cette modélisation nous permet aussi d'introduire facilement des gardes et des mises-à-jour des variables sur les arcs.

— La condition $v_a \geq x$ se modélise par la règle $x \cdot a \to x \cdot a$. Si on est capable de consommer x jetons dans a pour les restituer, cela signifie bien que nous possédons au moins x jetons dans a et donc que $v_a \geq x$.

— La condition $v_a \leq x$ est équivalente à la condition $a_{max} - v_{\bar{a}} \leq x$ qui est équivalent à la condition $v_{\bar{a}} \geq a_{max} - x$. Cette condition se modélise alors par la règle $(a_{max} - x) \cdot \bar{a} \to (a_{max} - x) \cdot \bar{a}$.

Ici encore nous n'ajoutons rien à l'expressivité des MSG étendus.

Nous illustrons à présent ces MSG étendus à l'aide de l'exemple suivant. Une seconde version simplifiée du protocole des fenêtres coulissantes est décrite sur la figure 7.6 sous la forme d'un MSG (à gauche) et d'un MSG étendu (à droite). Nous constatons à nouveau que la modélisation en PNS est plus concise et très simple à comprendre. Le compteur c correspond au nombre d'acquittements manquants. Lorsqu'un message est envoyé de l'instance i vers l'instance j, le compteur c est incrémenté et lorsque j envoie l'acquittement à i, le compteur c est décrémenté. De plus, des gardes sont placées sur les arcs de telle sorte que la valeur de c est toujours comprise entre 0 et 4.

Notons que les MSG sont généralement utilisés sous l'hypothèse de communications FIFO. Cette restriction peut également être considérée pour les MSG étendus de la manière suivante.

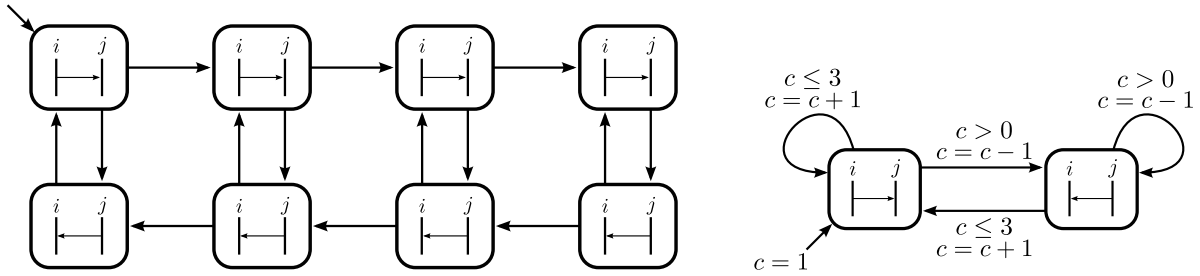


FIGURE 7.6 – Modélisation du protocole simplifié des fenêtres coulissantes avec un MSG à gauche et un PNS à droite.

Définition 7.3.4. *Un processus d'un MSG étendu est FIFO s'il est FIFO sur tous les canaux. Un processus d'un MSG étendu est FIFO sur un canal* (i, j) *si :*

— *Les messages ne se doublent pas dans ce canal.*

— *Les messages présents initialement dans ce canal sont consommés en premier.*

Remarquons que nous pouvons utiliser des compteurs afin de représenter des timers localisés. En effet, les timer peuvent être vus comme des variables initialisées avec une certaine valeur et décrémentée au cours du temps. En suivant la norme des MSC [92], les actions atomiques suivantes peuvent être effectuées :

— Un timer peut être initialisé : cela correspond à lui affecter une valeur.

— L'expiration d'un timer peut être vérifiée : cela correspond à effectuer un test à zéro.

— Un timer peut être arrêté : cela correspond à vérifier que le timer n'est pas expiré.

En considérant des timers discrets et bornés, les PNS peuvent effectuer simplement toutes ces actions de la manière suivante :

— Initialisation d'un timer sur l'instance i (figure 7.7(a)) :
$i \to i$ « $c_1 := 10$ ».

— Arrêt d'un timer avant son expiration, sur l'instance i (figure 7.7(a)) :
$i \to i$ « $c_1 > 0$ ».

— Constat de l'expiration d'un timer sur l'instance i (figure 7.7(c)) :
$i \to i$ « $c_1 == 0$ ».

Notons que nous pouvons également modéliser simplement des contraintes temporelles de délai localement sur une instance (figure 7.7(d)) :
$i \to i$ « $c_1 := 0$ » $i!j$
$i \to i$ « $2 \leq c_1 \leq 3$ » $i?j$.

Parallèlement à ces actions atomiques, la valeur de tous les timers doit pouvoir être décrémentée à tout moment. Afin de représenter l'écoulement du temps, pour chaque état du MSG, nous ajoutons des boucles implicites qui effectuent la décrémentation de tous les timers non expirés (figure 7.8).

Ben-Abdallah et Leue présentent dans [19] une version simplifiée du fonctionnement d'un distributeur automatique de billets (ATM) avec un MSG muni de contraintes tem-

(a) Initialisation (b) Arrêt (c) Expiration (d) Contrainte locale

FIGURE 7.7 – Représentation des timers dans les MSG étendus.

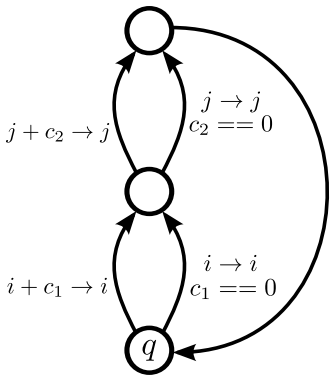


FIGURE 7.8 – Exemple d'une boucle implicite dans le cas d'un compteur c_1 localisé sur l'instance i et d'un compteur c_2 localisé sur l'instance j.

porelles mais sans compteur. Nous reprenons leur exemple en ne modélisant que la partie d'identification (figure 7.9) et en nous appuyant sur un compteur d'erreurs noté err.

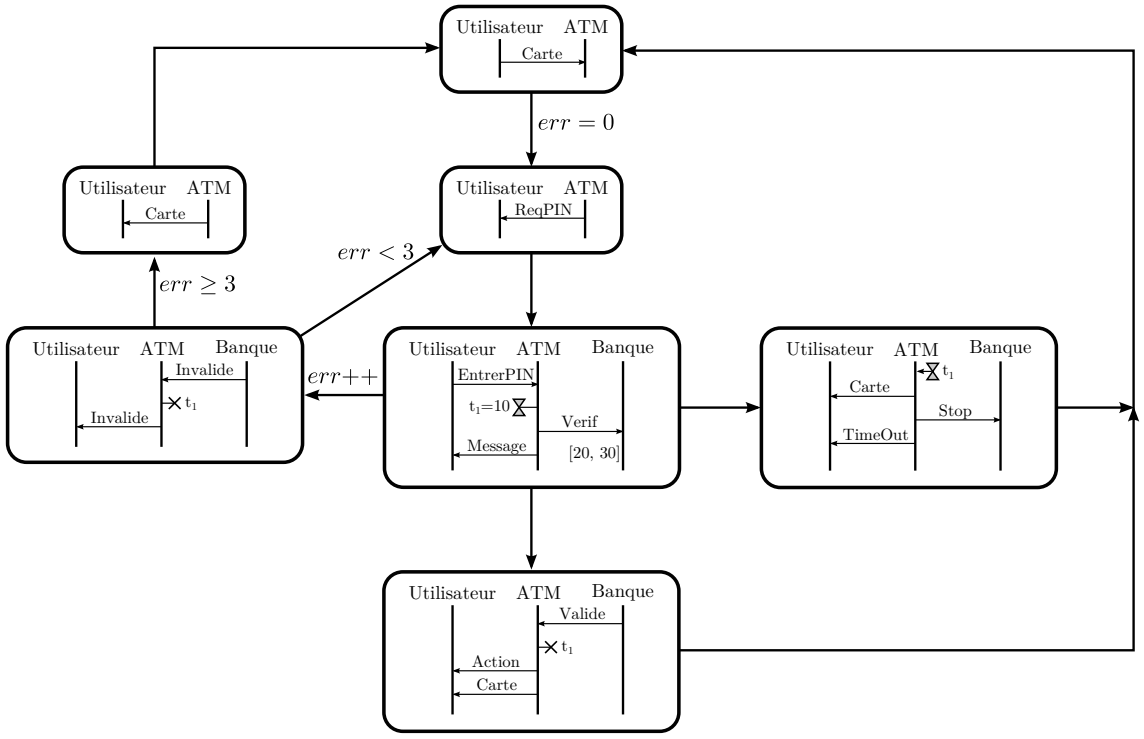


FIGURE 7.9 – Modélisation d'un distributeur automatique de billets (ATM) avec un MSG étendu.

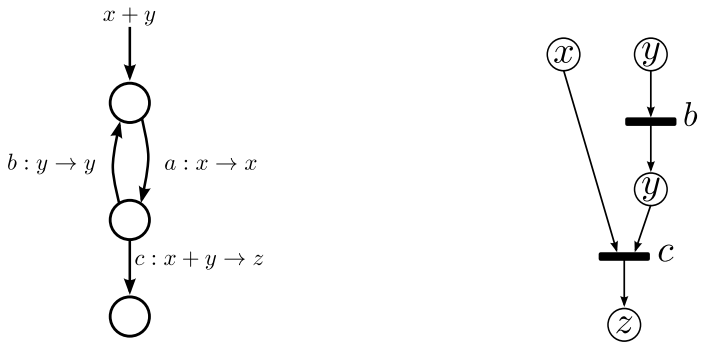
La première observation que nous pouvons faire est la simplification obtenue par l'ajout d'un compteur. En effet, contrairement à la modélisation de ce système en [19], nous n'avons pas besoin de recopier plusieurs fois des états identiques pour modéliser le fait qu'un utilisateur a le droit à trois essais uniquement.

Si nous vérifions l'accessibilité de l'état le plus en bas de ce MSG étendu, nous obtenons que cet état n'est pas accessible. En effet, la contrainte exigeant que la banque utilise au moins 20 unités de temps pour répondre implique que le timer t_1 expirera avant que la banque n'ait eu le temps de répondre. Si par contre nous réduisons cette contrainte en dessous de 10 unités de temps, alors cet état devient accessible.

Il faut souligner ici que le modèle ainsi obtenu permet de spécifier des protocoles temporisés sous une forme très différente de celle étudiée dans [8] et [48] car les contraintes temporelles sont formalisées à l'aide d'intervalles de temps entre les événements dans un MSC mais aussi entre les événements de deux MSC consécutifs sur une instance fixée. De plus le temps s'écoule de manière continue et peut s'écouler à des vitesses différentes sur les instances d'un même MSG, ce qui provoque le « drift » étudié dans [8]. Ce décalage permet de modéliser une machine à deux compteurs et conduit aux résultats d'indécidabilité de [48]. Cet aspect est néanmoins exclu de notre approche puisque le temps s'écoule de manière synchronisée à l'intérieur de chaque MSC.

7.4 Vérification des propriétés d'inclusion pour des réseaux bornés

Un problème classique en théorie de la concurrence consiste à caractériser le pouvoir d'expression d'un modèle. De plus, un problème couramment étudié est la synthèse d'un système à partir de sa spécification comportementale. Dans ce chapitre, nous considérons les réseaux de Petri avec états comme un moyen de spécifier des comportements parallèles sous la forme de processus. Nous abordons le problème consistant à construire un réseau de Petri équivalent à un réseau de Petri avec états. Deux classes de spécifications sont étudiées en fonction de la notion d'équivalence que nous adoptons.



(a) Un PNS non préfixe-réalisable (b) Un processus

FIGURE 7.10 – Un PNS et un processus non acceptant.

Définition 7.4.1. *Un réseau de Petri avec états $\mathcal{S}$ est* réalisable *(respectivement* préfixe-réalisable*) s'il existe un réseau de Petri $\mathcal{N}$ tel que $[\![\mathcal{S}]\!] = [\![\mathcal{N}]\!]$ (respectivement* $\mathrm{Pref}([\![\mathcal{S}]\!]) = [\![\mathcal{N}]\!]$*).*

Notons que le réseau de Petri avec états $\mathcal{S}$ de la figure 7.1(a) n'est *pas* réalisable, car l'ensemble des processus qu'il accepte n'est pas clos par préfixe (exemple 7.1.3) alors que l'ensemble des processus reconnus par un réseau de Petri est toujours clos par préfixe. Cependant, $\mathcal{S}$ est préfixe réalisable, car les préfixes de ces processus sont précisément les processus du réseau de Petri avec états muni d'un état unique et représenté sur la figure 2.5(a) (ou, de manière équivalente le réseau de Petri de la figure 2.5(b)). L'exemple suivant présente un réseau de Petri avec états qui n'est pas préfixe-réalisable.

Exemple 7.4.2. *Considérons le PNS $\mathcal{S}$ de la figure 7.10(a). Tout réseau de Petri $\mathcal{N}$ tel que* $[\![\mathcal{N}]\!] = \mathrm{Pref}([\![\mathcal{S}]\!])$ *accepterait le réseau causal $\mathcal{K}$ de la figure 7.10(b) en tant que processus. Cependant, $\mathcal{K}$ n'est évidement pas le préfixe d'un processus de $\mathcal{S}$, car la règle b ne peut pas avoir lieu sans la règle a. Ainsi $\mathcal{S}$* n'est pas *préfixe-réalisable.*

Bien que la réalisabilité semble être le problème le plus simple à considérer, nous affirmons que la préfixe-réalisabilité est aussi une question naturelle, car les processus d'un réseau de Petri sont clos par préfixe. L'observation suivante présente un candidat canonique pour la synthèse d'un réseau de Petri à partir d'un réseau de Petri avec états.

Propriété 7.4.3. *Soit $\mathcal{S}_1$ un réseau de Petri avec états et R_1 le sous-ensemble de règles se produisant dans les séquences de règles* tirables *de $\mathcal{S}_1$. Soit $\mathcal{S}_2$ un PNS muni d'un unique état et possédant le même marquage initial que $\mathcal{S}_1$ tel qu'une règle apparaît sur une boucle dans $\mathcal{S}_2$ si, et seulement si, elle appartient à R_1. Alors*

1. *$\mathcal{S}_1$ est réalisable si, et seulement, si $[\![\mathcal{S}_1]\!] = [\![\mathcal{S}_2]\!]$.*
2. *$\mathcal{S}_1$ est préfixe-réalisable si, et seulement, si* $\mathrm{Pref}([\![\mathcal{S}_1]\!]) = [\![\mathcal{S}_2]\!]$.

Démonstration. Supposons d'abord que $\mathcal{S}_1$ ne soit pas préfixe-réalisable. Alors $\mathrm{Pref}([\![\mathcal{S}_1]\!]) \neq [\![\mathcal{S}_2]\!]$ car $\mathcal{S}_2$ est équivalent à un réseau de Petri. Supposons maintenant que $\mathcal{S}_1$ soit préfixe-réalisable. Alors il existe un PNS $\mathcal{S}'$ avec un seul état tel que $\mathrm{Pref}([\![\mathcal{S}_1]\!]) = [\![\mathcal{S}']\!]$. Toute règle de R_1 apparaît dans un processus de $\mathcal{S}_1$, et donc dans un processus de $\mathcal{S}'$ et par conséquent, sur une boucle de $\mathcal{S}'$. Toute autre règle se produisant sur une boucle de $\mathcal{S}'$ ne peut pas se produire dans une séquence de règles tirable. Par conséquent, nous pouvons la retirer de $\mathcal{S}'$ sans affecter l'ensemble des processus de $\mathcal{S}'$. En d'autres termes, on peut supposer que $\mathcal{S}' = \mathcal{S}_2$. Un argument similaire vaut pour la réalisabilité. □

Notons que l'ensemble de règles R_1 peut être calculé à partir de $\mathcal{S}_1$ d'après la propriété 2.7.1. Il est clair que $\mathrm{Pref}([\![\mathcal{S}_1]\!]) \subseteq [\![\mathcal{S}_2]\!]$. Ainsi, la différence entre les spécifications de $\mathcal{S}_1$ et l'implémentation $\mathcal{S}_2$ découle des processus utilisant des règles de $\mathcal{S}_1$ qui ne sont pas représentées par des séquences de règles de $\mathcal{S}_1$. Cette situation est similaire à la notion de scénario implicite dans le cadre de la réalisabilité d'un MSG [9].

7.4.1 Un problème indécidable avec les traces de Mazurkiewicz

Le résultat d'indécidabilité présenté dans cette section s'appuie sur le problème d'universalité des traces de Mazurkiewicz [31] que nous rappelons ici. Soit Σ un alphabet fini d'actions. La concurrence dans un système distribué est souvent représentée par une *relation d'indépendance* sur Σ, qui est une relation binaire, symétrique, et irréflexive $\| \subseteq \Sigma \times \Sigma$. Alors, le couple $(\Sigma, \|)$ est appelé un *alphabet d'indépendance*. La *relation d'équivalence* associée est la plus petite congruence $\sim$ sur Σ^* telle que $a\|b$ implique $ab \sim ba$ pour tout $a, b \in \Sigma$. Nous notons par $[u]$ la classe d'équivalence d'un mot $u \in \Sigma^*$. Cette classe d'équivalence est appelée *trace* de u. Nous posons $[L] = \bigcup_{u \in L}[u]$ pour tout langage $L \subseteq \Sigma^*$.

Théorème 7.4.4. *[96, Théorème IV.4.3] Il est indécidable de savoir si $[L] = \Sigma^*$ pour un alphabet d'indépendance donné $(\Sigma, \|)$ et un langage régulier donné $L \subseteq \Sigma^*$.*

Puisque les PNS que nous considérons ne sont pas munis d'états acceptants, nous avons besoin du corollaire suivant.

Corollaire 7.4.5. *Il est indécidable de savoir si $[L] = \Sigma^*$ pour un alphabet d'indépendance donné $(\Sigma, \|)$ et un langage régulier* et clos par préfixe *donné $L \subseteq \Sigma^*$.*

Démonstration. Procédons par contradiction. Supposons que ce problème est décidable et montrons que le problème du théorème 7.4.4 devient décidable. Soit $(\Sigma, \|)$ un alphabet d'indépendance et $L \subseteq \Sigma^*$ un langage régulier. Considérons une lettre supplémentaire $\perp$ et le nouvel alphabet $\Gamma = \Sigma \cup \{\perp\}$ muni de la même relation d'indépendance : la nouvelle lettre $\perp$ est dépendante avec toutes les lettres de Σ. Soit $L' = \mathrm{Pref}(L) \cup (L \cdot \{\perp\} \cdot \Gamma^*)$. Il est clair que L' est régulier et clos par préfixe. De plus, $L \subseteq L'$. Pour conclure la preuve, nous pouvons vérifier facilement que $[L'] = \Gamma^*$ si, et seulement si, $[L] = \Sigma^*$.

Supposons d'abord que $[L] = \Sigma^*$. Il est clair que $[L'] \subseteq \Gamma^*$. Soit $v \in \Gamma^\star$. Nous distinguons deux cas. Si $v \in \Sigma^*$, alors $v \sim u$ pour un certain $u \in L$. Si $v \notin \Sigma^*$, alors $v = v_0.\perp.v_1$ avec $v_0 \in \Sigma^\star$ et $v_1 \in \Gamma^*$. En outre $v_0 \sim u_0$ pour $u_0 \in L$. Il en résulte que $v \sim u_0.\perp.v_1$ et $u_0.\perp.v_1 \in L'$. Dans les deux cas, nous obtenons que $v \in [u]$ pour $u \in L'$. Ainsi $[L'] = \Gamma^*$.

Inversement, supposons maintenant que $\Gamma^* = [L']$ et considérons $v \in \Sigma^*$. Alors $v.\perp \in \Gamma^\star$. Il existe un $u \in L'$ tel que $v.\perp \sim u$. Alors, $u = u_0.\perp$ car $\perp$ dépend de toutes les lettres. De plus, $v \sim u_0$ (car l'équivalence de trace est simplifiable à droite) et $u_0 \in L' \cap \Sigma^\star$. Il s'ensuit que $u_0 \in L$. Ainsi $[L] = \Sigma^\star$ □

Corollaire 7.4.6. *Soit $(\Sigma, \|)$ un alphabet d'indépendance. Il est indécidable de savoir si $[L_1] \subseteq [L_2]$ pour tout langage régulier et clos par préfixe $L_1, L_2 \subseteq \Sigma^*$.*

Démonstration. Considérons $L_1 = \Sigma^*$ et appliquons le corollaire 7.4.5. □

Dans la suite de ce chapitre, nous présentons un codage naturel des traces de Mazurkiewicz sous la forme d'un réseau causal. Alors, tout langage rationnel de traces de Mazurkiewicz clos par préfixe peut être représenté par un PNS borné par préfixe. Ainsi, nous obtiendrons que la relation d'inclusion $[\![\mathcal{S}_1]\!] \subseteq [\![\mathcal{S}_2]\!]$ est indécidable pour deux PNS bornés par préfixe $\mathcal{S}_1$ et $\mathcal{S}_2$.

7.4.2 Des traces de Mazurkiewicz aux processus

Soit $(\Sigma, \|)$ un alphabet d'indépendance fixé. Considérons un ensemble fini de places P et une application $\mathrm{Loc} : \Sigma \to 2^P$ telle que $a\|b$ si, et seulement si, $\mathrm{Loc}(a) \cap \mathrm{Loc}(b) = \emptyset$. Il y a plusieurs moyens de trouver un tel ensemble P muni d'une application Loc. Une manière de faire consiste à considérer tous les sous-ensembles $\{a, b\} \subseteq \Sigma$ tels que $a \not\| b$ comme une place ; puis à poser $\mathrm{Loc}(a) = \{p \in P \mid a \in p\}$. Il est clair que chaque place $p \in P$ apparaît dans la location $\mathrm{Loc}(a)$ d'une action $a \in \Sigma$. Posons $N = \Sigma$, $R_\Sigma = \{(a, \alpha, \alpha) \in R \mid \alpha = \mathrm{Loc}(a)\}$ et $\mu_{in} = P$. Notons qu'il y a exactement une règle $(a, \mathrm{Loc}(a), \mathrm{Loc}(a)) \in R_\Sigma$ pour chaque action $a \in \Sigma$. De plus, ces règles sont des *règles de synchronisation* selon la définition suivante.

Définition 7.4.7. *Une règle $r = (\lambda, \alpha, \beta)$ est une* règle de synchronisation *si $\alpha = \beta$ et $\alpha(p) \leq 1$ pour chaque $p \in P$.*

Ainsi, une règle de synchronisation consomme un jeton de certaines places et les restitue aussitôt.

Considérons l'application $\rho : \Sigma \to R_\Sigma$ telle que $\rho(a) = (a, \mathrm{Loc}(a), \mathrm{Loc}(a))$. Cette bijection s'étend naturellement à l'application des mots sur Σ vers les mots sur R_Σ.

Exemple 7.4.8. *Soit $\Sigma = \{a, b, c\}$ fourni avec la relation d'indépendance $a\|b$. Considérons $P = \{x, y\}$ avec $Loc(a) = \{x\}$, $Loc(b) = \{y\}$ et $Loc(c) = \{x, y\}$. La figure 7.11 représente un processus correspondant à la séquence de règles $\rho(abcab)$.*

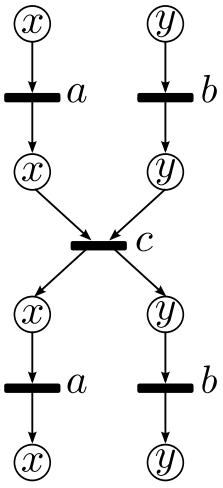


FIGURE 7.11 – Un processus représentant la séquence de règles $\rho(abcab)$ avec $a\|b$.

Notez que pour un mot $u \in \Sigma^\star$ la séquence de règles $\rho(u)$ est tirable à partir de μ_{in} et mène au marquage μ_{in}. Il résulte de la propriété 7.2.1 que tous les processus de $[\![\rho(u)]\!]_{\mu_{in}}$ sont isomorphes les uns des autres, c'est-à-dire intuitivement qu'il y a seulement un processus pour $\rho(u)$ à partir de μ_{in}.

Le résultat suivant affirme que des mots équivalents donnent lieu au même processus. Et inversement, si deux mots correspondent au même processus, alors ces deux mots sont dans la même classe d'équivalence. De cette façon, les classes d'équivalence de mots sont identifiées

avec des processus. Cette propriété est en fait similaire au fait bien connu que les traces de Mazurkiewicz peuvent être représentées par des ordres partiels étiquetés particuliers.

Lemme 7.4.9. *Pour tout $u, v \in \Sigma^*$: $u \sim v$ si, et seulement si, $[\![\rho(u)]\!]_{\mu_{in}} = [\![\rho(v)]\!]_{\mu_{in}}$.*

Démonstration. Soit $u \in \Sigma^*$ et $a, b \in \Sigma$ tel que $a \neq b$. Si $u.ab \sim u.ba$ alors $a \| b$, $\mathrm{Loc}(a) \cap \mathrm{Loc}(b) = \emptyset$, alors $[\![\rho(u.ab)]\!]_{\mu_{in}} = [\![\rho(u.ba)]\!]_{\mu_{in}}$ par la propriété 7.2.1. Ainsi $u \sim v$ implique $[\![\rho(u)]\!]_{\mu_{in}} = [\![\rho(v)]\!]_{\mu_{in}}$ pour tout $u, v \in \Sigma^\star$ (encore par la propriété 7.2.1). Pour prouver la propriété inverse, nous procédons par induction sur la longueur de u. Le cas de base est trivial. Considérons $u, v \in \Sigma^\star$ de longueur $n+1$ tel que $[\![\rho(u)]\!]_{\mu_{in}} = [\![\rho(v)]\!]_{\mu_{in}}$. Nous distinguons deux cas :

1. $u = u'.a$ et $v = v'.a$ pour tout $u', v' \in \Sigma^*$ et $a \in \Sigma$. Nous avons $\rho(u) = \rho(u').\rho(a)$ et $\rho(v) = \rho(v').\rho(a)$. Par la propriété 7.2.1 on a $[\![\rho(u')]\!]_{\mu_{in}} = [\![\rho(v')]\!]_{\mu_{in}}$. Il résulte de l'hypothèse d'induction que $u' \sim v'$ d'où $u'.a \sim v'.a$.
2. $u = u'.a$ et $v = v'.b$ pour $u', v' \in \Sigma^\star$ et $a, b \in \Sigma$ avec $a \neq b$. Nous avons $\rho(u) = \rho(u').\rho(a)$ et $\rho(v) = \rho(v').\rho(b)$ avec $\rho(a) \neq \rho(b)$. Alors tout réseau causal étiqueté $\mathcal{K}$ de $[\![\rho(u)]\!]_{\mu_{in}} = [\![\rho(v)]\!]_{\mu_{in}}$ contient deux événements maximaux e_a et e_b étiquetés par a et b respectivement. Il s'ensuit que $a \| b$. Soit $\mathcal{K}'$ le préfixe de $\mathcal{K}$ obtenu en retirant les deux événements maximaux e_a et e_b. Nous considérons une extension linéaire $w' \in \Sigma^*$ des événements de $\mathcal{K}'$. Alors $\mathcal{K}'$ est le processus de $[\![\rho(w')]\!]_{\mu_{in}}$. De plus, $[\![\rho(w'.a)]\!]_{\mu_{in}} = [\![\rho(v')]\!]_{\mu_{in}}$ et $[\![\rho(w'.b)]\!]_{\mu_{in}} = [\![\rho(u')]\!]_{\mu_{in}}$. Par hypothèse d'induction, nous obtenons $w'.a \sim v'$ et $w'.b \sim u'$. En revanche, $w'.ab \sim w'.ba$ car $a \| b$. Ainsi, $u \sim w'.ba \sim w'.ab \sim v$.

□

Nous considérons à présent un langage régulier et clos par préfixe $L \subseteq \Sigma^*$ et un automate fini $A(L) = (Q, \imath, \longrightarrow_{A(L)})$ dont les états sont tous acceptants et qui reconnaît L. Nous pouvons supposer que chaque état de $A(L)$ est accessible depuis l'état initial et que chaque action de Σ est utilisée au moins une fois par un arc étiqueté de $A(L)$. Nous construisons à partir de l'automate $A(L)$ le PNS $\mathcal{S}(L) = (Q, \imath, \longrightarrow_{\mathcal{S}(L)}, \mu_{in})$ avec le même ensemble d'états Q, le même état initial $\imath \in Q$ et tel que pour toute règle $r = (a, \alpha, \alpha) \in R_\Sigma$ et tout état $q_1, q_2 \in Q$, il existe un arc étiqueté $q_1 \xrightarrow{r}_{\mathcal{S}(L)} q_2$ si $q_1 \xrightarrow{a}_{A(L)} q_2$. Remarquons ici que le multi-ensemble de jetons reste inchangé par les règles. Par conséquent, l'ensemble des marquages accessibles par les préfixes de $[\![\mathcal{S}(L)]\!]$ est fini, c'est-à-dire que le PNS $\mathcal{S}_L$ est borné par préfixe. Pour tout langage régulier et clos par préfixe $L_1, L_2 \subseteq \Sigma^*$, le lemme 7.4.9 montre que nous avons $[L_1] \subseteq [L_2]$ si, et seulement si, $[\![\mathcal{S}_{L_1}]\!] \subseteq [\![\mathcal{S}_{L_2}]\!]$. Alors, le corollaire 7.4.6 permet d'affirmer que la propriété $[\![\mathcal{S}_1]\!] \subseteq [\![\mathcal{S}_2]\!]$ est indécidable pour deux PNS bornés par préfixes $\mathcal{S}_1$ et $\mathcal{S}_2$. Ce constat peut être renforcé par l'observation suivante.

Théorème 7.4.10. *Le problème de déterminer si un PNS borné par préfixe est réalisable est indécidable.*

Démonstration. Soit $L \subseteq \Sigma^*$ un langage régulier et clos par préfixe et $\mathcal{S}(L)$ le PNS correspondant. Soit $\mathcal{N}(L)$ le réseau de Petri collectant toutes les règles R_Σ de $\mathcal{S}(L)$. Alors

$[\![\mathcal{N}(L)]\!] = \bigcup_{u \in R_\Sigma^*} [\![u]\!]_{\mu_{in}}$. De plus, nous allons vérifier que $[\![\mathcal{S}(L)]\!] = [\![\mathcal{N}(L)]\!]$ si, et seulement si, $[L] = \Sigma^*$. Alors le corollaire 7.4.5 permet d'affirmer que $[\![\mathcal{N}(L)]\!] \subseteq [\![\mathcal{S}(L)]\!]$ est indécidable.

Supposons pour commencer que $[\![\mathcal{S}]\!] = [\![\mathcal{N}(L)]\!]$. Soit $u \in \Sigma^*$. Nous avons $[\![\rho(u)]\!]_{\mu_{in}} = [\![w]\!]_{\mu_{in}}$ avec $w \in CS(\mathcal{S})$. Soit $v = \rho^{-1}(w)$. Évidement, $v \in L$. Comme $[\![\rho(u)]\!]_{\mu_{in}} = [\![\rho(v)]\!]_{\mu_{in}}$, nous obtenons $u \sim v$ par le lemme 7.4.9. Ainsi, $\Sigma^* = [L]$.

Supposons maintenant que $[L] = \Sigma^*$. Soit $w \in R_\Sigma^*$. Nous avons $\rho^{-1}(w) \in \Sigma^*$. Alors $\rho^{-1}(w) \sim u$ avec $u \in L$. Il s'ensuit par le lemme 7.4.9 que $[\![w]\!]_{\mu_{in}} = [\![\rho(u)]\!]_{\mu_{in}}$. De plus, $\rho(u) \in CS(\mathcal{S})$. Par conséquent, $[\![\mathcal{N}(L)]\!] = [\![CS(\mathcal{S})]\!]_{\mu_{in}}$. □

Corollaire 7.4.11. *La propriété $[\![\mathcal{N}]\!] \subseteq [\![\mathcal{S}]\!]$ est indécidable pour un PNS borné par préfixe $\mathcal{S}$ et un réseau de Petri borné $\mathcal{N}$.*

Démonstration. Supposons par l'absurde que cette propriété est décidable. Montrons alors que nous pouvons décider si un PNS $\mathcal{S}$ borné par préfixe est réalisable. Soit $\mathcal{N}$ le réseau de Petri collectant toutes les règles accessibles dans $\mathcal{S}$. Nous avons $[\![\mathcal{N}]\!] \supseteq [\![\mathcal{S}]\!]$. Par la propriété 7.4.3, le PNS $\mathcal{S}$ est réalisable si et seulement si $[\![\mathcal{N}]\!] = [\![\mathcal{S}]\!]$. Deux possibilités :

— On a $[\![\mathcal{N}]\!] \subseteq [\![\mathcal{S}]\!]$, alors $[\![\mathcal{N}]\!] = [\![\mathcal{S}]\!]$ et donc $\mathcal{S}$ est réalisable.

— On a $[\![\mathcal{N}]\!] \not\subseteq [\![\mathcal{S}]\!]$, alors $[\![\mathcal{N}]\!] \neq [\![\mathcal{S}]\!]$ et donc $\mathcal{S}$ n'est pas réalisable.

Ainsi, la décidabilité de cette propriété impliquerait la décidabilité du théorème 7.4.10. □

La séparation entre les PNS et les réseaux de Petri est illustrée par l'observation suivantes. Le problème de décision du corollaire 7.4.11 est décidable lorsque l'on se limite aux réseaux de Petri, et même si l'on considère des réseaux de Petri non-bornés.

Propriété 7.4.12. *Soit $\mathcal{N}_1$ et $\mathcal{N}_2$ deux réseaux de Petri. La propriété $[\![\mathcal{N}_1]\!] \subseteq [\![\mathcal{N}_2]\!]$ est décidable.*

Démonstration. Observons d'abord que cette propriété exige que $\mathcal{N}_1$ et $\mathcal{N}_2$ partagent le même marquage initial. Soit R_i l'ensemble des règles qui se produisent dans une séquence de règles tirable de $\mathcal{N}_i$. L'ensemble R_i peut être effectivement calculé (propriété 2.7.1). Alors $[\![\mathcal{N}_1]\!] \subseteq [\![\mathcal{N}_2]\!]$ si, et seulement si, $R_1 \subseteq R_2$. □

Chapitre

8 Vérification de propriétés MSO sur les processus d'un réseau borné

La sémantique des processus des réseaux de Petri peut être utilisée pour modéliser et vérifier des systèmes avec des contraintes comportementales, telles que les canaux FIFO, des communications causales, des clés privées, ou toute restriction exprimable en MSO. Dans ce chapitre, nous étudions le problème consistant à vérifier si tous les processus d'un réseau de Petri avec états $\mathcal{S}$ satisfont une formule ψ exprimée en logique monadique du second ordre (MSO), c'est-à-dire que chaque réseau causal de $[\![\mathcal{S}]\!]$ est un modèle de ψ, ce que l'on note $\mathcal{S} \models \psi$. Nous présentons une technique permettant d'établir que ce problème est décidable (théorème 8.2.4). Ce résultat généralise l'étude du model-checking de formules MSO sur les MSG. À notre connaissance, ce problème de model-checking n'a pas encore été abordé, même dans le cas particulier des réseaux de Petri. Nous montrons que ce problème est décidable. Notre approche est relativement simple et repose sur le théorème de Büchi [27] qui montre l'équivalence entre la définissabilité d'un ensemble de mots par une formule MSO et son caractère reconnaissable par un automate fini de mots.

Le model-checking des graphes représentant les exécutions d'un système sur les formules MSO a été étudié dans différents contextes. Si la classe des graphes considérés est définissable en logique MSO et tree-width bornée, la satisfiabilité d'une formule MSO est connue pour être décidable [33], [98], [78]. Cependant les processus d'un PNS ne sont pas nécessairement MSO-définissables — même dans le cas particulier d'un MSG non-divergent, car les MSG non divergents peuvent décrire des ensembles non réguliers de MSC. Ainsi ces travaux ne s'appliquent pas à notre problématique. D'autre part, la classe des processus de tout réseau de Petri borné est MSO-définissable. Par conséquent, l'ordre partiel des événements peut être décrit par un automate concurrent ou une structure d'événement régulière [102] pour lesquels le model-checking est possible [57]. Cependant, cette approche ne s'applique pas pour les réseaux de Petri avec états s'ils ne sont pas préfixe bornés ou si leurs processus ne sont pas MSO-définissables.

La technique présentée dans ce chapitre nous permet de vérifier efficacement des propriétés MSO sur les processus d'un PNS borné (pas nécessairement préfixe borné). Puisque les comportements FIFO peuvent être caractérisés par des sentences MSO, ce résultat étend le résultat principal de [77] qui affirme que le problème de model-checking pour les MSG est décidable (voir aussi [50, Cor 6.1]). Contrairement à [50, 77], nous ne supposons pas que tous les comportements sont FIFO (puisque cette notion n'a pas de sens pour les PNS en général) et par conséquent nous ne pouvons pas faire usage de la notion de *linéarisations représentatives*. Le fait est que, comme déjà mentionné, une séquence de règles peut correspondre à plusieurs processus non isomorphes en fonction de l'ordre dans lequel les particules identiques sont consommées. C'est la principale différence avec les MSC, car ceux-ci sont complètement spécifiés par l'une de leurs linéarisations.

8.1 Logique MSO

Dans ce chapitre, nous fixons un PNS borné $\mathcal{S}$ avec un marquage initial μ_{in} sur un ensemble fini de places P et un ensemble fini de règles R. Afin de simplifier la présentation de nos résultats, nous considérons que les événements d'un processus sont étiquetés par une règle plutôt que par un nom de règle. Il ne s'agit là nullement d'une restriction puisque la règle correspondant à un événement d'un processus se déduit aisément du processus. La logique MSO que nous considérons s'applique à la classe des ordres partiels dont les nœuds sont étiquetés par des lettres de l'union disjointe $\Sigma = P \dot{\cup} R$, qui comprend notamment les processus de chaque séquence de règles $s \in R^\star$. Ainsi, les modèles que nous considérons ici sont des triplets $(N, \preceq, \lambda)$ où N est un ensemble fini de nœuds, $\preceq$ est un ordre partiel sur N, et λ est une application de N vers $\Sigma = P \dot{\cup} R$.

Les formules de la logique MSO que nous considérons utilisent des variables du premier ordre $x, y, z...$ pour les nœuds et des variables du second ordre $X, Y, Z...$ pour les ensembles de nœuds. Elles sont construites à partir des formules atomiques $P_a(x)$ pour $a \in \Sigma$ (qui signifie « le nœud x est étiqueté par la lettre a »), $x \preceq y$, et $x \in X$ par l'intermédiaire des connecteurs booléens $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ et les quantificateurs $\exists, \forall$ (pour les variables du premier et second ordre). Les formules sans variables libres sont appelées *sentences*.

La relation $\models$ entre un ordre partiel étiqueté $(N, \preceq, \lambda)$ et une formule est définie canoniquement en sachant que les variables du premier ordre portent sur des nœuds de N et les variables du second ordre portent sur les sous-ensembles de N. La classe des ordres partiels étiquetés qui satisfont une formule φ est désignée par $\mathrm{Mod}(\varphi)$. On dit qu'une classe d'ordres partiels étiquetés $\mathcal{L}$ est *MSO-definable* s'il existe une sentence φ telle que $\mathcal{L} = \mathrm{Mod}(\varphi)$.

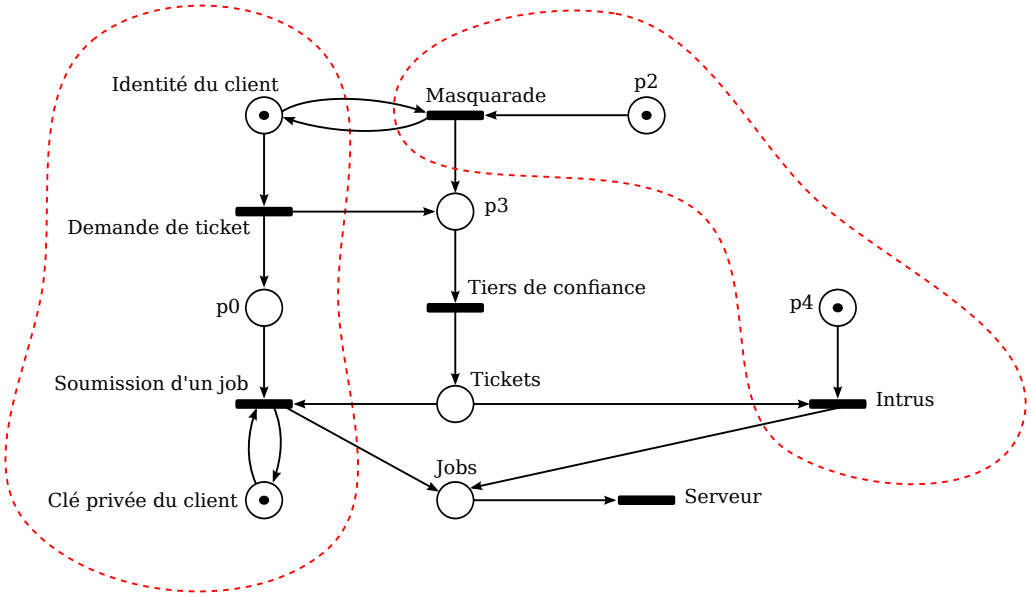


FIGURE 8.1 – Un protocole cryptographique simple.

Exemple 8.1.1. *Le réseau de Petri de la figure 8.1 décrit un protocole cryptographique simple pour la soumission de travaux à un serveur. Le client est spécifié sur le côté gauche*

de la figure. Il peut demander des tickets à un tiers de confiance à l'aide de sa propre identité. Ensuite, le tiers produit un ticket qui peut être utilisé pour soumettre un travail au serveur, à l'aide de la clé privée du client. Le comportement d'un intrus est représenté sur le côté droit. Il peut utiliser l'identité du client pour produire une demande de ticket ou intercepter des tickets. Considérons à présent les trois propriétés suivantes :

P1. *Un ticket ne peut être consommé sans la clé privée du client.*

P2. *Le serveur ne consomme pas de travaux soumis par l'intrus.*

P3. *Le client consomme seulement des tickets qu'il a demandé.*

Ces propriétés peuvent être facilement formalisées par des formules MSO sur les processus :

P1. $\forall x, (x : Tickets) \rightarrow (x : ClePrivee)$

P2. $Serveur \hookleftarrow Jobs \hookleftarrow SoumissionDeJob$

P3. $SoumissionDeJob \hookleftarrow Tickets \hookleftarrow TiersDeConfiance \hookleftarrow p3 \hookleftarrow DemandeDeTicket$

Nous utilisons deux symboles syntaxiques pour simplifier l'écriture des formules. Le premier symbole correspond aux deux-points, par exemple $x : Tickets$*. Cela signifie que* x *est une transition qui consomme un jeton de la place* $Tickets$*. Le deuxième symbole correspond à la flèche vers la gauche, par exemple* $Serveur \hookleftarrow Jobs \hookleftarrow SoumissionDeJob$ *et alterne entre les noms de transition et les noms de place. Cela signifie que* $Serveur$ *consomme un jeton de* $Jobs$ *qui a été produit par* $SoumissionDeJob$*.*

Nous aimerions vérifier que (P1)→(P2) pour tous les processus du réseau de Petri. De plus, il serait intéressant de calculer un contre-exemple (sous la forme d'un processus) pour la propriété (P1)→(P3).

8.2 Comment décider la propriété $\mathcal{S} \models \psi$?

Afin de résoudre le problème de model-checking considéré, l'approche que nous adoptons consiste à travailler sur des mots qui représentent une séquence de règles tirable et qui sont une extension linéaire de chaque processus de cette séquence de règle. Afin de vérifier que tous processus de toute séquence de règles tirable de $\mathcal{S}$ satisfait la propriété ψ donnée, il faut pouvoir reconstruire les processus à partir du mot représentatif, c'est-à-dire formaliser les relations de causalité entre les conditions et les événements dans les processus. Puisqu'une séquence de règles peut correspondre à plusieurs processus, plusieurs possibilités doivent être considérées. Nous introduisons la notion de coloriage de processus à l'aide de partitions de l'ensemble des nœuds d'un mot représentatif afin de caractériser les processus d'une séquence de règle. Nous expliciterons comment ces coloriages déterminent un processus et peuvent être formalisés dans la logique MSO.

Comme $\mathcal{S}$ est borné, on peut calculer et fixer un nombre naturel $\mathcal{B}$ tel que chaque marquage accessible μ de $\mathcal{S}$ est $\mathcal{B}$-borné, c'est-à-dire, $\mu(p) \leq \mathcal{B}$ pour chaque $p \in P$. Une séquence de règles $s = r_1...r_m \in R^\star$ tirable à partir de μ_{in} est dite $\mathcal{B}$-bornée si le marquage atteint par chaque sous-séquence $r_1...r_l$ est $\mathcal{B}$-borné. En particulier, toute séquence de règles tirable dans $\mathcal{S}$ est $\mathcal{B}$-bornée.

Nous fixons un mot $w_{\text{in}} \in P^\star$ qui est une extension linéaire du multi-ensemble μ_{in}, c'est-à-dire que $|w_{\text{in}}|_p = \mu_{in}(p)$ pour tout $p \in P$. De même, pour chaque règle $r \in R$, nous fixons un mot $w_r = r.w'_r$ où $|w'_r|_p = r^\bullet(p)$ pour tout $p \in P$. Ensuite, pour chaque séquence de règles $s = r_1...r_m$ tirable de $\mathcal{S}$, la séquence $w_s = w_{\text{in}}.w_{r_1}...w_{r_m}$ est appelée le *mot représentatif de* s. Nous considérons w_s comme un ensemble totalement ordonné de nœuds étiquetés par des lettres de Σ et nous posons $w_s = (N, \leq, \lambda)$ où N est l'ensemble de nœuds, $\leq$ est l'ordre total sur N, et $\lambda : N \to \Sigma$ est l'étiquetage correspondant.

Les nœuds étiquetés par une place sont appelés des *nœuds-places* alors que les nœuds étiquetés par des règles sont appelés des *nœuds-règles*. Il est intéressant de voir que w_s peut être regardé comme une extension linéaire d'un processus de s, où les nœuds-places qui suivent un nœud-règle étiqueté par r correspond au multi-ensemble de jetons $r^\bullet$ produit par cette occurrence de r. Puisque $\mathcal{S}$ est borné, l'ensemble des séquences de règles tirables dans $\mathcal{S}$ est régulier et par conséquent l'ensemble des mots représentatifs de ses séquences de règles aussi.

Afin de construire un processus de s à partir du mot représentatif w_s, nous avons besoin de spécifier quels jetons disponibles sont consommés par chaque occurrence de règle. Pour ce faire, nous utilisons une coloration des nœuds-places de w_s de sorte qu'à chaque étape, tous les jetons disponibles dans une place donnée possèdent des couleurs distinctes. De plus, nous munissons également les nœuds-règles avec une série d'autres couleurs afin de préciser quels jetons sont consommés à chaque étape de s. Ceci conduit à la notion de coloriage de processus.

Définition 8.2.1. *Soit $w = (N, \leq, \lambda)$ un ordre linéaire des nœuds étiquetés par Σ. Un* coloriage de processus *de w se compose*

- *d'une* partition $C = \{C_1, ..., C_{\mathcal{B}}\}$ *de l'ensemble de nœuds-places ; un nœud-place $n \in N$ est dit être coloré par k dans la place p si $\lambda(n) = p$ et $n \in C_k$.*
- *pour chaque place $p \in P$ et chaque $k \in [1..\mathcal{B}]$, d'un sous-ensemble de nœuds-règles $D_{p,k}$; nous disons qu'un nœud-règle $n \in N$ consomme un jeton de couleur k dans la place p si $n \in D_{p,k}$.*

De plus, les trois conditions suivantes doivent être remplies :

PC1 *Pour chaque nœud-règle n, pour chaque place $p \in P$, nous avons $\#\{k \in [1..\mathcal{B}] \mid n \in D_{p,k}\} = ({}^\bullet\lambda(n))(p)$;*

PC2 *Pour chaque place $p \in P$ et chaque couleur $k \in [1..\mathcal{B}]$, chaque couple de nœuds-places colorées par k dans la place p est séparée par un nœud-règle qui consomme un jeton de couleur k dans la place p ;*

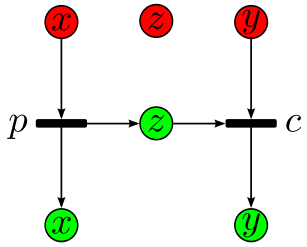
PC3 *Pour chaque nœud règle n qui consomme un jeton coloré par k dans la place p, il existe un nœud-place précédent $n' \leq n$ coloré par k dans la place p tel qu'aucun nœud-règle entre n' et n ne consomme des jetons de couleur k dans la place p.*

Intuitivement, un nœud-place appartient à C_k s'il décrit un jeton de couleur k dans la place $\lambda(n) \in P$. Un nœud-règle n appartient à $D_{p,k}$ s'il décrit une occurrence de la règle $\lambda(n) \in R$ qui consomme un jeton coloré par k dans la place p. Ainsi, la condition

PC1 affirme que n consomme le multi-ensemble approprié de jetons dans chaque place, à condition que ces jetons aient des couleurs distinctes. Précisément, **PC2** garantit que les couleurs données aux nouveaux jetons produits dans une place par l'exécution d'une règle diffèrent des couleurs utilisées par les jetons disponibles dans cette place. Elle assure également que les jetons produits dans une place par l'exécution d'une règle prennent des couleurs distinctes. Par conséquent, à chaque étape, tous les jetons disponibles dans une place ont des couleurs distinctes. Pour finir de déterminer la causalité dans un processus de s à partir d'un coloriage de processus de w_s, nous devons nous assurer qu'il y a suffisamment de jetons colorés disponibles lorsque chaque règle est appliquée. La dernière exigence **PC3** garantit que pour chaque nœud-règle qui consomme un jeton de couleur k dans la place p, un jeton de ce genre apparaît avant la règle et n'a pas été encore consommé.

w_s	x	y	z	p	z	x	c	y
C_1	✗		✗					✗
C_2		✗			✗	✗		
$D_{x,1}$				✗				
$D_{y,2}$							✗	
$D_{z,2}$							✗	

(a) Un coloriage de processus de $w_s = xyzpzxcy$



(b) Le processus correspondant

FIGURE 8.2 – Exemple de coloriage de processus.

Nous allons établir que la notion de coloration de processus caractérise précisément les extensions linéaires des processus et permet de construire un processus de s à partir du mot w_s. Nous montrons tout d'abord que tout coloriage de processus détermine un processus (propriété 8.2.2) ; puis que tout processus correspond à un coloriage (propriété 8.2.3). Considérons par exemple la séquence de règles $s = pc$ partant du marquage initial $\mu_{in} = \{x, y, z\}$ où $p : x \to x + z$ et $c : y + z \to y$. Un coloriage de processus de $w_s = xyzpzxcy$ avec $\mathcal{B} = 2$ est donné par le tableau de la figure 8.2(a). Le processus correspondant est représenté dans la figure 8.2(b).

Propriété 8.2.2. *Soit $w_s = (N, \leq, \lambda)$ un ordre linéaire de nœuds étiquetés par Σ qui correspond au mot représentatif d'une séquence de règles $s \in R^\star$. Soit $C = (C_k)_{k\in[1..\mathcal{B}]}$ et $D = (D_{p,k})_{p\in P, k\in[1..\mathcal{B}]}$ un coloriage de processus de w_s. Soit $\prec\!\!\!\cdot$ la relation binaire sur N tel que $x \prec\!\!\!\cdot y$ si*

- *ou bien x est un nœud-règle suivi du nœud-place y avec aucun nœud-règle entre les deux.*
- *ou bien y est un nœud-règle précédé d'un nœud-place x coloré par k dans la place p tel que y consomme un jeton coloré par k dans la place p et sans qu'il y ait de nœud-règle entre x et y consommant un jeton coloré par k dans la place p.*

Soit $\preceq$ la fermeture réflexive et transitive de $\prec\!\!\!\cdot$. Alors l'ordre étiqueté partiel $(N, \preceq, \lambda)$ est un processus de s tirable à partir de μ_{in}, noté $\mathcal{K}_{C,D}(s)$. De plus, s est $\mathcal{B}$-borné.

Démonstration. On procède par récurrence sur la longueur de s. Le cas de base où $|s| = 0$ est trivial, car $\prec\!\!\!-$ est vide et $\mu_{in}(p) \leq \mathcal{B}$ pour chaque $p \in P$. Étape d'induction. Soit $s = r_1...r_m$ une séquence de règles de longueur $m \geq 1$ et $w_s = (N, \leq, \lambda)$ son mot représentatif. Soit (C, D) un coloriage de processus de w_s. Nous considérons la sous-séquence $s' = r_1...r_{m-1}$ et son mot représentatif $w_{s'} = (N', \leq, \lambda)$ avec $N' \subsetneq N$. Il est facile de vérifier que la restriction de la coloration de processus (C, D) pour les nœuds de N' est une coloration de processus de $w_{s'}$. Soit $\prec\!\!\!-_{s'}$ la relation binaire sur N' et $\preceq_{s'}$ l'ordre partiel associé. Par hypothèse d'induction, $(N', \preceq_{s'}, \lambda)$ est un processus $\mathcal{K}'$ de s' et s' est une séquence de règles tirable partant de μ_{in} et $\mathcal{B}$-bornée. Notons que la relation binaire $\prec\!\!\!-$ est acyclique. Par ailleurs, la restriction de $(N, \preceq, \lambda)$ aux nœuds de N' est précisément le processus $\mathcal{K}' = (N', \preceq_{s'}, \lambda)$ de s'. Nous avons besoin de vérifier que l'ajout des nœuds de $N \setminus N'$ à $\mathcal{K}'$ selon $\prec\!\!\!-$ produit un processus de s. Pour ce faire, nous vérifions les trois propriétés suivantes :

1. Pour chaque place $p \in P$, le nœud-règle $n^\circ \in N \setminus N'$ correspondant à r_k couvre au plus ${}^\bullet r_k(p)$ nœuds-places étiquetés par p, c'est-à-dire que $\#\{n \in N' \mid n \prec\!\!\!- n^\circ \wedge \lambda(n) = p\} \leq {}^\bullet r_k(p)$. Cela découle de **PC1** et **PC2**.
2. Pour chaque place $p \in P$, le nœud-règle $n^\circ \in N \setminus N'$ correspondant à r_k couvre au moins ${}^\bullet r_k(p)$ nœuds-places étiquetés par p, c'est-à-dire que $\#\{n \in N' \mid n \prec\!\!\!- n^\circ \wedge \lambda(n) = p\} \geq {}^\bullet r_k(p)$. Cela découle de **PC1** et **PC3**.
3. Les conditions ne sont pas branchées, c'est-à-dire que pour chaque nœud-place $n \in N$, si $n \prec\!\!\!- n_1$ et $n \prec\!\!\!- n_2$ alors $n_1 = n_2$. Ceci est assuré par la définition même de $\prec\!\!\!-$.

Il s'ensuit que l'ordre partiel étiqueté $\mathcal{K} = (N, \preceq, \lambda)$ est un processus de s. Ainsi s est tirable à partir de μ_{in}. Pour conclure, pour chaque place p et chaque couleur k, **PC2** garantit qu'au plus un nœud-place coloré par k dans la place p n'est pas couvert par un nœud règle. Par conséquent, le marquage μ atteint par le processus $\mathcal{K}$ satisfait $\mu(p) \leq \mathcal{B}$ pour tout $p \in P$. Ainsi s est $\mathcal{B}$-borné. □

Ainsi, chaque coloriage de processus w_s donne un processus de $[\![s]\!]_{\mu_{in}}$. Par conséquent s est tirable à partir de μ_{in} lorsque w_s admet une coloration de processus. Alors, s doit être $\mathcal{B}$-bornée. Inversement, le résultat suivant affirme que chaque processus d'une séquence de règles s, tirable à partir de μ_{in}, peut être obtenu par une coloration de processus de w_s, à condition que s soit $\mathcal{B}$-bornée.

Propriété 8.2.3. *Soit $s = r_1...r_m$ une séquence de règles $\mathcal{B}$-bornée tirable à partir de μ_{in} et $\mathcal{K}$ un processus de s. Alors, il existe une coloration de processus (C, D) du mot représentatif w_s tel que $\mathcal{K}_{C,D}(s)$ est isomorphe à $\mathcal{K}$.*

Démonstration. Nous procédons par récurrence sur la longueur de s. Le cas de base où $|s| = 0$ est trivial, car μ_{in} est $\mathcal{B}$-borné. Étape d'induction. Soit $\mathcal{K} = (B \cup E, F^*, \lambda)$ un processus de $s = r_1...r_m$ de longueur $m \geq 1$ avec un ensemble de conditions B et un ensemble d'événements E. Nous considérons la sous-séquence $s' = r_1...r_{m-1}$ et le préfixe $\mathcal{K}' = (B' \cup E', F^*, \lambda)$ de $\mathcal{K}$ dans lequel l'événement $e^\circ \in E$ correspondant à la dernière occurrence de r_m et les conditions suivantes sont supprimées. Clairement, $\mathcal{K}'$ est un processus de s'. Par hypothèse d'induction, il existe une coloration de processus (C', D') de $w_{s'}$ telle

que $\mathcal{K}_{C',D'}(s') = (N', \preceq_{C',D'}, \lambda)$ est isomorphe à $\mathcal{K}' = (B' \cup E', F, \lambda)$. Nous notons $\sigma : B' \cup E' \to N'$ un isomorphisme de $\mathcal{K}'$ vers $\mathcal{K}_{C',D'}(s')$, c'est-à-dire une bijection telle que

— $x_1 F^* x_2$ si, et seulement si, $\sigma(x_1) \preceq_{C',D'} \sigma(x_2)$ pour tout $x_1, x_2 \in B' \cup E'$ et

— $\lambda(\sigma(x)) = \lambda(x)$ pour tout $x \in B' \cup E'$.

Nous étendons la bijection $\sigma : B' \cup E' \to N'$ à une bijection $\sigma : B \cup E \to N$ telle que

— $\sigma(e^\circ)$ est le nœud-règle de $N \setminus N'$, et

— chaque condition c de $B \setminus B'$ est mise en correspondance avec un nœud-place $\sigma(c) \in N \setminus N'$ tel que $\lambda(\sigma(c)) = \lambda(c)$.

Nous étendons également la coloration de processus (C', D') à la coloration de processus (C, D) de w_s en deux étapes :

1. Pour toutes les places $p \in P$ et pour toutes les couleurs $k \in [1..\mathcal{B}]$, le nœud-règle $\sigma(e^\circ)$ appartient à $D_{p,k}$ si l'événement e° couvre une condition c telle que le nœud correspondant $\sigma(c)$ est étiqueté par p et coloré par k, c'est-à-dire que $\lambda(c) = p$ et $\sigma(c) \in C'_k$.
2. Les couleurs des places-nœuds additionnelles de $N \setminus N'$ sont choisies de sorte que toutes les conditions maximales de $\mathcal{K}$ étiquetées par une même place p ont des couleurs distinctes. Ceci est possible, car le marquage atteint par s est $\mathcal{B}$-borné.

On peut vérifier que la coloration résultante (C, D) est une coloration de processus.

PC1 Soit $p \in P$ et $n^\circ = \sigma(e^\circ)$. Comme $\mathcal{K}$ est un processus, $\#\{k \in [1..\mathcal{B}] \mid n^\circ \in D_{p,k}\} \leq ({}^\bullet\lambda(n^\circ))(p)$. On peut vérifier que toutes les conditions étiquetées par p et couvertes par l'événement e° ont des couleurs distinctes, en raison de **PC2**. Ainsi, $\#\{k \in [1..\mathcal{B}] \mid n^\circ \in D_{p,k}\} = ({}^\bullet\lambda(n^\circ))(p)$.

PC2 La propriété requise est valable pour tout couple de nœuds-places de $w_{s'}$ car (C', D') est un coloriage de processus. Ceci est également vrai pour tout couple de nœuds-places de $N \setminus N'$. Si un nœud-place appartient à N' et un nœud-place appartient à $N \setminus N'$, la propriété reste vraie, car un nœud-règle n° apparaît entre les deux nœuds-places.

PC3 La propriété requise est vraie pour chaque nœud-règle $n \in N'$. Soit $p \in P$ et $k \in [1..\mathcal{B}]$ tels que $n^\circ \in D_{p,k}$. Par définition de $D_{p,k}$, l'événement e° couvre une condition c telle que le nœud correspondant $n = \sigma(c)$ est étiqueté par p et coloré par k. De plus, il n'y a pas de nœud-règle n' entre n et n° avec $n' \in D_{p,k}$. Autrement, la condition c serait également couverte par un événement dans $\mathcal{K}'$, donc c serait une condition avec un branchement dans $\mathcal{K}$.

Rappelons que σ met en correspondance chaque condition de $B \setminus B'$ vers un nœud-place de $N \setminus N'$ avec la même étiquette. Toutes ces conditions portent sur e° et tous ces nœuds-places couvrent $\sigma(e^\circ)$. Pour conclure, nous avons besoin de vérifier que pour chaque nœud-place $n \in N'$, nous avons $n \prec\!\!\!\!- _{C,D} n^\circ$ dans $\mathcal{K}_{C,D}(s)$ si, et seulement si, $\sigma^{-1}(n) F e^\circ$ dans $\mathcal{K}$.

Supposons d'abord que $n \prec\!\!\!\!- _{C,D} n^\circ$. Alors il existe une place $p \in P$ et une couleur $k \in [1..\mathcal{B}]$ telles que $\lambda(n) = p$, $n \in C_k$, et $n^\circ \in D_{p,k}$. En outre, il existe une condition c dans le

processus $\mathcal{K}$ telle que cFe°, $\lambda(c) = p$ et $\sigma(c) \in C_k$. Alors $\sigma(c) = n$ en raison de **PC2**. Ainsi $\sigma^{-1}(n)Fe^\circ$ dans $\mathcal{K}$.

Réciproquement, supposons maintenant que $\sigma^{-1}(n)Fe^\circ$ dans $\mathcal{K}$. Soit $p = \lambda(n)$ et k la couleur telle que $n \in C_k$. Nous avons $n^\circ \in D_{p,k}$ d'après la définition de $D_{p,k}$. Aucun événement $\mathcal{K}'$ ne couvre $\sigma^{-1}(n)$ donc il n'existe pas de nœud-règle dans $w_{s'}$ après n qui est coloré par $D_{p,k}$. Il s'ensuit que $n \prec\!\!\!\!- _{C,D} n^\circ$. □

Ainsi, la notion de coloration de processus caractérise les processus de toute séquence de règles $\mathcal{B}$-bornée tirable à partir de μ_{in}.

Suivant le théorème de Büchi, nous pouvons concevoir une formule MSO $\phi_{\mathcal{S}}$ qui définit les mots $w = (N, \leq, \lambda)$ sur Σ qui sont des mots représentatifs d'une séquence de règles $\mathcal{S}$. Nous pouvons également concevoir une formule $\phi_{pc}(C, D)$ avec $\mathcal{B} \times (|P| + 1)$ variables libres du second ordre $C = (C_k)_{k\in[1..\mathcal{B}]}$ et $D = (D_{k,p})_{k\in[1..\mathcal{B}],p\in P}$ qui caractérise la notion d'une coloration de processus pour un mot $w = (N, \leq, \lambda)$ sur Σ. Par ailleurs, au moyen de la propriété 8.2.2, nous pouvons construire une formule $\phi_{\preceq}(x, y, C, D)$ avec deux variables libres du premier ordre x, y et $\mathcal{B} \times (|P| + 1)$ variables libres du second ordre telle que pour toute interprétation de $C = (C_k)_{k\in[1..\mathcal{B}]}$, $D = (D_{k,p})_{k\in[1..\mathcal{B}],p\in P}$ et toute interprétation de x et y, $\phi_{\preceq}(x, y, C, D)$ est satisfaite si, et seulement si, nous avons $x \preceq y$ dans le processus correspondant à la coloration du processus donnée par l'interprétation.

Soit ψ une sentence MSO pour l'ordre partiel étiqueté sur Σ. Nous considérons la formule suivante $\bar{\psi}_{\mathcal{S}}$ pour les mots sur Σ :

$$\bar{\psi}_{\mathcal{S}} = \phi_{\mathcal{S}} \wedge \exists C, \exists D, (\phi_{pc}(C, D) \wedge \neg\psi'(C, D))$$

où la formule $\psi'(C, D)$ est obtenue à partir de ψ en remplaçant chaque occurrence de $x \preceq y$ par $\phi_{\preceq}(x, y, C, D)$. Il est clair qu'un mot satisfait $\bar{\psi}_{\mathcal{S}}$ si, et seulement si, il s'agit d'un mot représentatif d'une séquence de règles s de $\mathcal{S}$ pour laquelle il existe une coloration de processus qui décrit un processus satisfaisant $\neg\psi$. De cette façon, nous obtenons le résultat principal de ce chapitre.

Théorème 8.2.4. *Soit $\mathcal{S}$ un PNS borné et ψ une sentence MSO sur les réseaux causaux. Tous les processus de $\mathcal{S}$ satisfont ψ si, et seulement si, la sentence $\bar{\psi}_{\mathcal{S}}$ n'est pas satisfiable.*

8.3 Mise en œuvre

Ainsi, le problème du model-checking pour un PNS et une formule MSO est décidable. Dans la pratique, l'insatisfiabilité de $\bar{\psi}_{\mathcal{S}}$ est réduit au problème du vide d'un automate fini. Il est bien sûr plus efficace de ne pas inclure la sentence $\phi_{\mathcal{S}}$ et de comparer l'automate résultant avec l'automate qui reconnaît les mots représentatifs des séquences de règles de $\mathcal{S}$. Il convient de noter que nous pourrions munir le PNS $\mathcal{S}$ d'un sous-ensemble d'états acceptants et vérifier que tous les processus issus d'une séquence de règles acceptantes satisfont ψ.

Nous avons implémenté cette technique en nous appuyant sur l'outil MONA [3] qui permet de résoudre des formules MSO et l'outil TINA [4] qui permet de modéliser et vérifier divers propriétés sur les réseaux de Petri. Notre prototype [5] nous permet notamment de

spécifier d'abord un réseau de Petri avec TINA en lui demandant une borne supérieure des marquages accessibles. Puis d'appliquer le théorème 8.2.4 pour vérifier une formule MSO sur les processus avec l'aide de MONA.
Continuons l'exemple 8.1.1 ; nous avons pu vérifier que (P1)→(P2) pour tous les processus du réseau de Petri. De plus, notre prototype a été en mesure de calculer un contre-exemple sous la forme d'un processus (figure 8.3) pour (P1)→(P3).

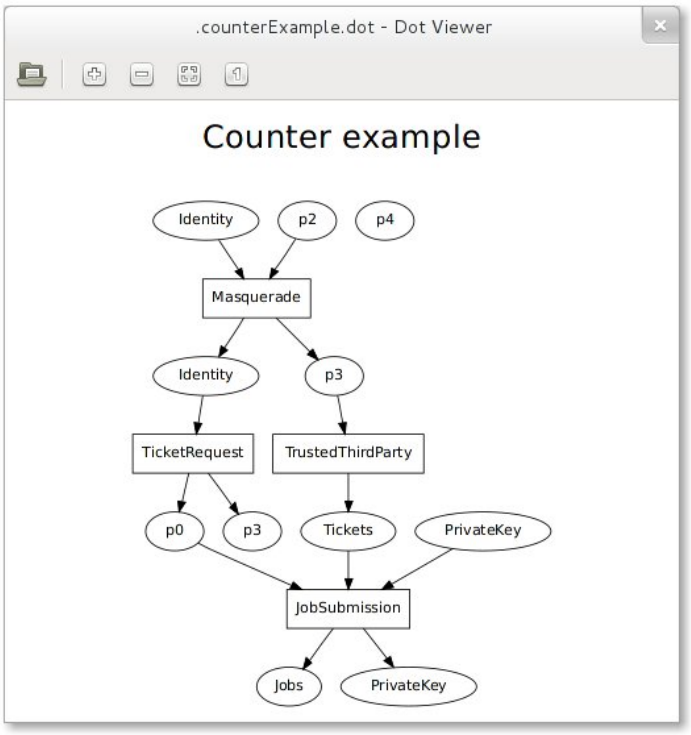


FIGURE 8.3 – Contre exemple pour (P1)→(P3).

Nous voyons dans le contre exemple de la figure 8.3 un processus pour lequel le client consomme un ticket qu'il n'a pas demandé alors que le ticket est consommé avec la clé privée du client. Ce processus ne satisfait donc pas la formule (P1)→(P3).

Notons que les comportements FIFO peuvent être facilement caractérisés en MSO lorsque le PNS modélise un système qui échange des messages. Ainsi, la décidabilité de MSO vaut également pour des modélisations utilisant une sémantique FIFO pour les MSG étendus. Il est toutefois préférable de considérer ce comportement directement dans les coloriages ce qui simplifierait la vérification de la formule.

Chapitre

9 Vérification de propriétés d'accessibilité des préfixes

Les problèmes de décision classiques sur l'ensemble des marquages accessibles d'un réseau de Petri sont connus pour être décidables, à savoir le caractère borné (définition 2.3.5), la couverture (définition 2.3.6) et l'accessibilité (définition 2.3.4). Nous avons vu, dans la sous section 2.7, une simulation d'un PNS par un réseau de Petri qui représente les mêmes marquages accessibles. De ce fait, tous les résultats classiques sur l'ensemble des marquages accessibles s'appliquent aux PNS.

Cependant, l'adoption d'une sémantique d'ordres partiels nous amène de nouvelles difficultés car le modèle des PNS n'est plus équivalent aux réseaux de Petri. Par exemple, la question $[\![\mathcal{S}_1]\!] = [\![\mathcal{S}_2]\!]$ est

— *décidable* si $\mathcal{S}_1$ et $\mathcal{S}_2$ sont deux réseaux de Petri (propriété 7.4.12).

— *indécidable* si $\mathcal{S}_1$ et $\mathcal{S}_2$ sont deux PNS (même s'ils sont bornés par préfixe) comme nous l'avons établi au corollaire 7.4.11.

Dans ce chapitre, nous nous intéressons à trois problèmes de vérification classiques sur l'ensemble des marquages accessibles par les *préfixes* des processus : le caractère borné, la couverture et l'accessibilité. Nous montrons comment réduire ces problèmes au cas particulier des réseaux de Petri de telle sorte que tous les résultats de complexité et de décidabilité s'étendent des réseaux de Petri au PNS sous la sémantique des processus.

Définition 9.0.1. *Un marquage μ est* accessible par préfixe *dans un PNS $\mathcal{S}$ s'il existe un préfixe d'un processus de $\mathcal{S}$ qui conduit au marquage μ.*

Ainsi, tout marquage accessible est un marquage accessible par préfixe. Cependant, l'ensemble des marquages accessibles par préfixe peut être distinct de l'ensemble des marquages accessibles. Par exemple, chaque processus du PNS de la figure 7.1(a) conduit à un marquage avec au plus 3 jetons alors que les préfixes de ces processus conduisent à une infinité de marquages distincts (on peut voir sur la figure 7.1(b) un préfixe d'un processus qui conduit à un marquage avec 4 jetons). Dans le cas particulier des réseaux de Petri, cependant, tout marquage accessible par préfixe est accessible, car la classe des processus est close par préfixe. Ainsi, les problèmes que nous étudions dans ce chapitre sont connus pour les réseaux de Petri, mais nouveaux pour les réseaux de Petri avec états.

Le premier problème que nous considérons est le problème du caractère borné par préfixe, qui consiste à savoir si l'ensemble des marquages accessibles par un préfixe d'un PNS donné $\mathcal{S}$ est fini. Nous proposons dans ce chapitre une construction linéaire d'un PNS $\mathcal{S}^\circ$ à partir de $\mathcal{S}$ telle que $\mathcal{S}$ est borné par préfixe si, et seulement si, $\mathcal{S}^\circ$ est borné. Puisque le caractère borné de $\mathcal{S}^\circ$ se réduit au caractère borné d'un réseau de Petri, on obtient que le problème du caractère borné par préfixe d'un PNS est équivalent au problème du caractère borné d'un

réseau de Petri (théorème 9.3.1). En outre, nous montrons que cette technique s'applique aux deux autres problèmes sur les marquages accessibles, à savoir la couverture par préfixe et l'accessibilité par préfixe (théorème 9.3.2 et 9.3.3).

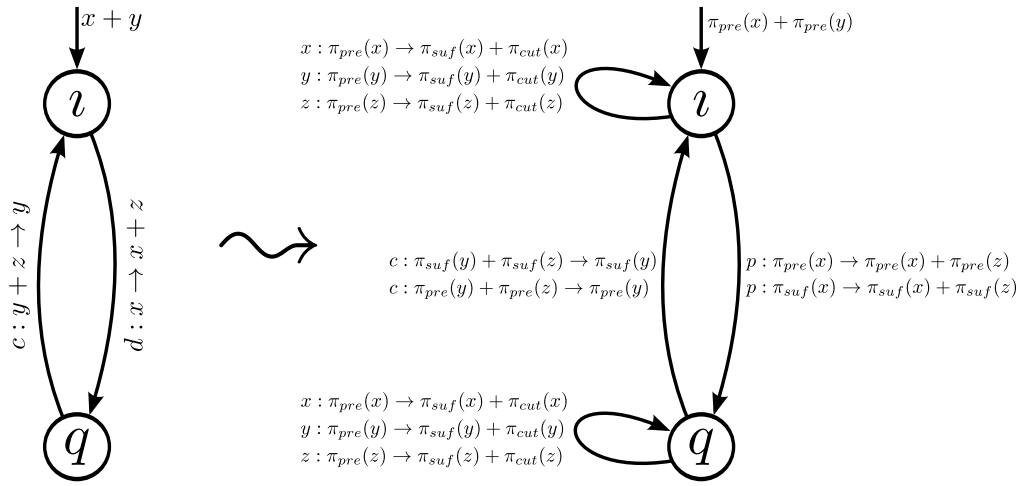


FIGURE 9.1 – Vérification d'un marquage préfixe accessible par préfixe.

Pour terminer, nous montrons que dans le cas d'un MSG étendu, si un marquage est préfixe accessible alors ce marquage est préfixe accessible sous l'hypothèse FIFO. Cela permet d'étendre tous les résultats sur les marquages préfixes accessibles au cas des MSG étendus.

9.1 D'un PNS à un réseau de Petri

Soit $\mathcal{S} = (Q, \imath, \longrightarrow, \mu_{in})$ un PNS fixé. Nous construisons un PNS $\mathcal{S}^\circ$ qui nous permet d'analyser l'ensemble des marquages accessibles par les préfixes de $\mathcal{S}$. La construction de $\mathcal{S}^\circ$ à partir de $\mathcal{S}$ est illustrée sur la figure 9.1, où le PNS $\mathcal{S}^\circ$ résultant du PNS $\mathcal{S}$ de la figure 7.1(a) est représenté. Le PNS $\mathcal{S}^\circ$ permet l'utilisation de trois ensembles disjoints de places : $P_{\text{pre}}, P_{\text{suf}}, P_{\text{cut}}$ qui sont des copies de l'ensemble des places P de $\mathcal{S}$. Nous notons $\pi_{\text{pre}} : P \to P_{\text{pre}}$, $\pi_{\text{suf}} : P \to P_{\text{suf}}$, et $\pi_{\text{cut}} : P \to P_{\text{cut}}$ des bijections qui mettent en correspondance chaque place de P vers la place correspondante dans P_{pre}, P_{cut} et P_{suf} respectivement. Ces applications s'étendent naturellement aux applications de multi-ensembles vers des multi-ensembles. Le marquage initial ${\mu_{in}}^\circ$ de $\mathcal{S}^\circ$ est le multi-ensemble ${\mu_{in}}^\circ = \pi_{\text{pre}}(\mu_{in})$.

Le PNS $\mathcal{S}^\circ$ partage avec $\mathcal{S}$ l'ensemble de ses états Q et son état initial $\imath$. Il se compose de trois ensembles disjoints d'arcs étiquetés : $\longrightarrow_{\text{pre}}, \longrightarrow_{\text{suf}}, \longrightarrow_{\text{cut}}$. La restriction de $\mathcal{S}^\circ$ aux arcs étiquetés $\longrightarrow_{\text{pre}}$ donne un PNS ${\mathcal{S}^\circ}_{\text{pre}}$ isomorphe à $\mathcal{S}$. Ainsi, pour chaque arc étiqueté $q_1 \xrightarrow{r} q_2$ dans $\mathcal{S}$ avec $r = (a, {}^\bullet r, r^\bullet)$, il existe un arc étiqueté $q_1 \xrightarrow{s}_{\text{pre}} q_2$ avec $s = (a, \pi_{\text{pre}}({}^\bullet r), \pi_{\text{pre}}(r^\bullet))$. De même, la restriction de $\mathcal{S}^\circ$ aux arcs étiquetés $\longrightarrow_{\text{suf}}$ donne un PNS P_{suf} isomorphe à $\mathcal{S}$, si ce n'est que son marquage initial est vide : pour chaque arc étiqueté $q_1 \xrightarrow{r} q_2$ dans $\mathcal{S}$ avec $r = (a, {}^\bullet r, r^\bullet)$, il existe un arc étiqueté $q_1 \xrightarrow{s}_{\text{suf}} q_2$ avec $s = (a, \pi_{\text{suf}}({}^\bullet r), \pi_{\text{suf}}(r^\bullet))$. Intuitivement les deux PNS ${\mathcal{S}^\circ}_{\text{pre}}$ et ${\mathcal{S}^\circ}_{\text{suf}}$ sont synchronisés, car ils partagent un même ensemble d'états. L'ensemble des arcs étiquetés $\longrightarrow_{\text{cut}}$ se compose d'une boucle $q \xrightarrow{s}_{\text{cut}} q$ pour chaque état q et chaque place $p \in P$; cet arc étiqueté permet de déplacer un jeton

de la place $\pi_{\text{pre}}(p)$ à la place $\pi_{\text{suf}}(p)$ et de garder une trace de ce transfert dans la place $\pi_{\text{cut}}(p)$, c'est-à-dire que ${}^\bullet s = \pi_{\text{pre}}(p)$ et $s^\bullet = \pi_{\text{suf}}(p) + \pi_{\text{cut}}(p)$. Notons que les jetons de P_{cut} ne peuvent jamais être consommés. Notons également que si (λ, α, β) est une règle, alors $\pi_{\text{suf}}(\lambda, \alpha, \beta) = (\lambda, \pi_{\text{suf}}(\alpha), \pi_{\text{suf}}(\beta))$, et $\pi_{\text{pre}}(\lambda, \alpha, \beta) = (\lambda, \pi_{\text{pre}}(\alpha), \pi_{\text{pre}}(\beta))$

Intuitivement, pour tout processus $\mathcal{K}$ de $\mathcal{S}$ et pour tout préfixe $\mathcal{K}'$ de $\mathcal{K}$, le PNS $\mathcal{S}^\circ$ peut simuler une séquence de règles de $\mathcal{S}$ qui correspond à $\mathcal{K}$ de telle sorte que chaque événement du préfixe $\mathcal{K}'$ correspond à l'emprunt d'un arc étiqueté de $\longrightarrow_{\text{pre}}$ et chaque événement du suffixe $\mathcal{K} \setminus \mathcal{K}'$ correspond à l'emprunt d'un arc étiqueté de $\longrightarrow_{\text{suf}}$. Par ailleurs l'ensemble des places P_{cut} garde la trace des jetons transférés de $\mathcal{K}$ vers $\mathcal{K}'$, c'est-à-dire de ${\mathcal{S}^\circ}_{\text{pre}}$ vers ${\mathcal{S}^\circ}_{\text{suf}}$, par des arcs étiquetés de $\longrightarrow_{\text{cut}}$. Ainsi tout marquage accessible par préfixe de $\mathcal{S}$ est représenté par la restriction à $P_{\text{pre}} \cup P_{\text{cut}}$ d'un marquage accessible de $\mathcal{S}^\circ$. La propriété essentielle de cette représentation, énoncée dans la propriété 9.1.1 ci-dessous, affirme que réciproquement chaque séquence de règles tirable de $\mathcal{S}^\circ$ correspond à un processus $\mathcal{K}$ de $\mathcal{S}$ et un préfixe $\mathcal{K}'$ de $\mathcal{K}$ telles que le marquage de $P_{\text{pre}} \cup P_{\text{cut}}$ décrit le marquage atteint par $\mathcal{K}'$.

Afin de prouver nos résultats dans les détails, nous adopterons les notations suivantes :

— On note par τ_{pre} et τ_{suf} les bijections qui associent à chaque arc étiqueté $q \xrightarrow{r} q'$ de $\mathcal{S}$ l'arc étiqueté correspondant dans $\longrightarrow_{\text{pre}}$ et $\longrightarrow_{\text{suf}}$ respectivement.

— Pour chaque état $q \in Q$, on note τ^q_{cut} la bijection entre P et l'ensemble des boucles $q \longrightarrow_{\text{cut}} q$ associées à chaque place p.

Dans la suite de ce chapitre, pour chaque marquage μ et pour chaque sous-ensemble de places X, nous notons par $\mu|X$ la restriction de μ aux places de X. Les principaux résultats de ce chapitre reposent essentiellement sur l'observation suivante. Nous affirmons que tout marquage accessible par préfixe de $\mathcal{S}$ est représenté par un marquage accessible de μ° et vice versa.

Propriété 9.1.1. *Un multi-ensemble de places $\mu \in \mathbb{N}^P$ est accessible par préfixe dans $\mathcal{S}$ si, et seulement si, il existe un marquage accessible μ° de $\mathcal{S}^\circ$ tel que $\mu = \pi^{-1}_{pre}(\mu^\circ|P_{pre}) + \pi^{-1}_{cut}(\mu^\circ|P_{cut})$.*

9.2 Preuve de la propriété 9.1.1

Soit $\mathcal{S} = (Q, \imath, \longrightarrow, \mu_{in})$ un PNS. Pour chaque séquence de règles tirable $u = r_1 ... r_n \in R^\star$ à partir de μ_{in}, nous notons μ_u le marquage accessible par u depuis μ_{in}, c'est-à-dire que $\mu_u = \mu_{in} + \sum_{i=1}^{n} (r_i{}^\bullet - {}^\bullet r_i)$. De même, pour chaque séquence de transition tirable $s \in T^\star_{\mathcal{N}}$ depuis le marquage initial ${\mu_{in}}^\circ$, ${\mu^\circ}_s$ désigne le marquage atteint par s dans $\mathcal{S}^\circ$.

Nous utiliserons la notion suivante d'exécution partielle : une *exécution partielle* est un triplet $(u, v, w) \in R^\star \times R^\star \times R^\star$ tels que $[\![v.w]\!]_{\mu_{in}} \cap [\![u]\!]_{\mu_{in}} \neq \emptyset$ et $u \in CS(\mathcal{S})$. Alors $[\![v]\!]_{\mu_{in}} \neq \emptyset$ et donc la séquence de règles v est tirable à partir de μ_{in}. Une exécution partielle est utilisée en tant que témoin assurant l'existence d'un préfixe $\mathcal{K}_v \in [\![v]\!]_{\mu_{in}}$ d'un processus $\mathcal{K}_u \in [\![u]\!]_{\mu_{in}}$. Notons que v n'a pas besoin d'être un préfixe de u, ni d'être une séquence de règles de $\mathcal{S}$. Les

exécutions partielles sont étroitement liées à l'accessibilité des préfixes, comme le montre l'observation suivante.

Propriété 9.2.1. *Pour chaque exécution partielle (u, v, w), le marquage μ_v est accessible par préfixe. Inversement, pour tout marquage μ accessible par préfixe, il existe une exécution partielle (u, v, w) telle que $\mu = \mu_v$.*

Démonstration. Soit (u, v, w) une exécution partielle. Alors il existe un réseau causal $\mathcal{K}$ tel que $\mathcal{K} \in [\![v.w]\!]_{\mu_{in}} \cap [\![u]\!]_{\mu_{in}}$. Par la propriété 7.2.1, $\mathcal{K}$ peut être construit à partir d'un réseau causal $\mathcal{K}_v \in [\![v]\!]_{\mu_{in}}$ en ajoutant la séquence de règles w, les unes après les autres. Ainsi, $\mathcal{K}_v$ est un préfixe de $\mathcal{K}$ et μ_v est accessible par préfixe.

Soit $\mathcal{K} \in [\![u]\!]_{\mu_{in}}$ avec $u \in CS(\mathcal{S})$ et $\mathcal{K}'$ un préfixe de $\mathcal{K}$. Soit v une extension linéaire de l'ordre partiel des règles qui se produisent dans $\mathcal{K}'$. Alors $\mathcal{K}' \in [\![v]\!]_{\mu_{in}}$ et μ_v est le marquage accessible par $\mathcal{K}'$. Soit w une extension linéaire de l'ordre partiel des règles présentes dans le suffixe $\mathcal{K} \setminus \mathcal{K}'$. Alors $v.w$ est une extension linéaire de l'ordre partiel des règles qui se produisent dans $\mathcal{K}$ et donc $\mathcal{K} \in [\![v.w]\!]_{\mu_{in}}$. Par conséquent, (u, v, w) est une exécution partielle. □

Lemme 9.2.2. *Soit (u, v, w) une exécution partielle et $a \in R$ une règle telle que ${}^\bullet a \leq \mu_u$. Si $u.a \in CS(\mathcal{S})$ alors $(u.a, v, w.a)$ est une exécution partielle.*

Démonstration. Soit $\mathcal{K}$ un réseau causal étiqueté de $[\![v.w]\!]_{\mu_{in}} \cap [\![u]\!]_{\mu_{in}}$. Comme ${}^\bullet a \leq \mu_u$, la classe $[\![u.a]\!]_{\mu_{in}}$ n'est pas vide (propriété 7.2.1). De plus, tous les réseaux causaux de $[\![u.a]\!]_{\mu_{in}}$ sont obtenus à partir d'un réseau causal de $[\![u]\!]_{\mu_{in}}$ en ajoutant une occurrence de la règle a. En particulier, nous pouvons ajouter une occurrence de la règle a à $\mathcal{K}$ et obtenir un réseau causal de $[\![u.a]\!]_{\mu_{in}}$. Ce dernier est aussi un réseau causal de $[\![v.w.a]\!]_{\mu_{in}}$ car $\mathcal{K} \in [\![v.w]\!]_{\mu_{in}}$. □

La preuve de la propriété 9.1.1 s'appuie sur les deux prochains lemmes qui peuvent être établis au moyen d'inductions un peu fastidieuses. La première affirme que pour chaque séquence de règles tirable $u \in \mathrm{FCS}(\mathcal{S})$ et chaque préfixe $\mathcal{K}_v$ de chaque processus $\mathcal{K}_u \in [\![u]\!]_{\mu_{in}}$, le PNS $\mathcal{S}^\circ$ peut être guidé afin de simuler chaque règle de u dans son ordre séquentiel afin que les marquages atteints par u soient décrits par le marquage courant de $P_{\mathrm{pre}} \cup P_{\mathrm{suf}}$, tandis que le marquage atteint par $\mathcal{K}_v$ est décrit par le marquage courant de $P_{\mathrm{pre}} \cup P_{\mathrm{cut}}$. De plus, nous devons faire en sorte que l'état $q \in Q$ atteint par u soit également accessible par s dans $\mathcal{S}^\circ$ et vérifier que tous les événements de $\mathcal{K}_u$ qui ne se produiront pas dans $\mathcal{K}_v$ soient effectués par des transitions de $\longrightarrow_{\mathrm{suf}}$. Pour ce faire, nous devons guider $\mathcal{S}^\circ$ pour transférer exactement le nombre requis de jetons de P_{pre} vers P_{suf}, ce qui correspond au marquage de P_{cut}.

Lemme 9.2.3. *Soit (u, v, w) une exécution partielle de $\mathcal{S}$ et q un état tel que $\imath \xrightarrow{u} q$ dans $\mathcal{S}$. Il existe une séquence de règles tirable s dans $\mathcal{S}^\circ$ telle que*

1. $\pi^{-1}_{cut}(\mu^\circ{}_s | P_{cut}) + \pi^{-1}_{pre}(\mu^\circ{}_s | P_{pre}) = \mu_v$,
2. $\pi^{-1}_{suf}(\mu^\circ{}_s | P_{suf}) + \pi^{-1}_{pre}(\mu^\circ{}_s | P_{pre}) = \mu_u$,
3. $\pi^{-1}_{cut}(\mu^\circ{}_s | P_{cut}) = req(w)$,

4. $\imath \xrightarrow{s} q$ *dans* $\mathcal{S}^\circ$.

Démonstration. Nous procédons par induction sur la longueur de u. Si $|u| = 0$ alors $u = \varepsilon$, $q = \imath$, $v = \varepsilon$ et $w = \varepsilon$. La séquence de règles vide de $\mathcal{S}^\circ$ satisfait aux quatre propriétés du lemme, car ${\mu_{in}}^\circ = \pi_{\text{pre}}(\mu_{in})$. Étape d'induction : nous considérons une exécution partielle (u', v', w') avec $|u'| = n+1$. Soit $u' = u.a$ avec $|u| = n$ et $\imath \xrightarrow{u} q \xrightarrow{a} q'$ dans $\mathcal{S}$. Soit $\mathcal{K}_{u'}$, $\mathcal{K}_{v'}$ et $\mathcal{K}_{w'}$ trois réseaux causaux tels que $\mathcal{K}_{u'} \in [\![u']\!]_{\mu_{in}} \cap [\![v'.w']\!]_{\mu_{in}}$, $\mathcal{K}_{v'} \in [\![v']\!]_{\mu_{in}}$ est un préfixe de $\mathcal{K}_{u'}$ et $\mathcal{K}_{w'} \in [\![w']\!]_{\mu_{v'}}$ est le suffixe correspondant. Nous savons que $\mathcal{K}_{u'}$ est obtenu à partir d'un processus $\mathcal{K}_u \in [\![u]\!]_{\mu_{in}}$ en ajoutant l'événement e_a qui correspond à l'occurrence de la règle a. Nous distinguons deux cas.

1. L'événement e_a apparaît dans $\mathcal{K}_{w'}$. Alors il existe une séquence de règles w telle que $\mathcal{K}_{w'} \in [\![w.a]\!]_{\mu_{v'}}$ et $\mathcal{K}_u \in [\![v'.w]\!]_{\mu_{in}}$, donc (u, v', w) est une exécution partielle de longueur n. Par hypothèse de récurrence, il existe une séquence de règles tirable s dans $\mathcal{S}^\circ$ telle que les quatre propriétés du lemme sont satisfaites. En particulier, $\pi^{-1}_{\text{cut}}({\mu^\circ}_s|P_{\text{cut}}) = \text{req}(w)$. De plus, la propriété 7.2.3 assure que $\text{req}(w.a) \leq \mu_{v'}$ car $\mathcal{K}_{w'} \in [\![w.a]\!]_{\mu_{v'}}$. De plus, nous avons d'une part $\pi^{-1}_{\text{cut}}({\mu^\circ}_s|P_{\text{cut}}) = \text{req}(w) \leq \text{req}(w.a)$; et d'autre part $\text{req}(w.a) \leq \mu_{v'}$, c'est-à-dire $\text{req}(w.a) \leq \pi^{-1}_{\text{cut}}({\mu^\circ}_s|P_{\text{cut}}) + \pi^{-1}_{\text{pre}}({\mu^\circ}_s|P_{\text{pre}})$. Ainsi,

$$\pi^{-1}_{\text{cut}}({\mu^\circ}_s|P_{\text{cut}}) \leq \text{req}(w.a) \leq \pi^{-1}_{\text{cut}}({\mu^\circ}_s|P_{\text{cut}}) + \pi^{-1}_{\text{pre}}({\mu^\circ}_s|P_{\text{pre}})$$

Il s'ensuit qu'il existe un multi-ensemble de places $\hat{\mu} \in \mathbb{N}^P$ tel que $\pi^{-1}_{\text{cut}}({\mu^\circ}_s|P_{\text{cut}}) + \hat{\mu} = \text{req}(w.a)$ avec $\pi_{\text{pre}}(\hat{\mu}) \leq {\mu^\circ}_s|P_{\text{pre}}$. Nous considérons une séquence de transitions $A \in \longrightarrow^\star_{\text{cut}}$ dans $\mathcal{S}^\circ$ qui consomme le multi-ensemble de jetons $\pi_{\text{pre}}(\hat{\mu})$ et produit le multi-ensemble de jetons $\pi_{\text{suf}}(\hat{\mu})$. Ainsi, ${\mu^\circ}_{s.A}|P_{\text{suf}} = {\mu^\circ}_s|P_{\text{suf}} + \pi_{\text{suf}}(\hat{\mu}) \geq {}^\bullet a$. Nous avons

$${}^\bullet a + \mu_v - \mu_u = {}^\bullet a + \text{cost}(w) \leq \max(\text{req}(w), {}^\bullet a + \text{cost}(w)) = \text{req}(w.a)$$

Ainsi $\mu_u - \mu_v + \text{req}(w.a) \geq {}^\bullet a$. D'autre part,

$$\mu_u - \mu_v + \text{req}(w.a) = \mu_u - \mu_v + \pi^{-1}_{\text{cut}}({\mu^\circ}_s|P_{\text{cut}}) + \hat{\mu} = \mu_u - \pi^{-1}_{\text{pre}}({\mu^\circ}_s|P_{\text{pre}}) + \hat{\mu} = \pi^{-1}_{\text{suf}}({\mu^\circ}_s|P_{\text{suf}}) + \hat{\mu}$$

Ainsi, $\pi^{-1}_{\text{suf}}({\mu^\circ}_{s.A}|P_{\text{suf}}) = \pi^{-1}_{\text{suf}}({\mu^\circ}_s|P_{\text{suf}}) + \hat{\mu} \geq {}^\bullet a$. Il s'ensuit que $s.A.\tau_{\text{suf}}(a)$ est une séquence de règles tirable $\mathcal{S}^\circ$. Celle-ci remplit les conditions du lemme :

   (a) $\pi^{-1}_{\text{pre}}({\mu^\circ}_{s.A.\tau_{\text{suf}}(a)}|P_{\text{pre}}) = \pi^{-1}_{\text{pre}}({\mu^\circ}_s|P_{\text{pre}}) - \hat{\mu}$ et $\pi^{-1}_{\text{cut}}({\mu^\circ}_{s.A.\tau_{\text{suf}}(a)}|P_{\text{cut}}) = \pi^{-1}_{\text{cut}}({\mu^\circ}_s|P_{\text{cut}}) + \hat{\mu}$ d'où $\pi^{-1}_{\text{pre}}({\mu^\circ}_{s.A.\tau_{\text{suf}}(a)}|P_{\text{pre}}) + \pi^{-1}_{\text{cut}}({\mu^\circ}_{s.A.\tau_{\text{suf}}(a)}|P_{\text{cut}}) = \pi^{-1}_{\text{pre}}({\mu^\circ}_s|P_{\text{pre}}) + \pi^{-1}_{\text{cut}}({\mu^\circ}_s|P_{\text{cut}}) = \mu_{v'}$.

   (b) $\pi^{-1}_{\text{pre}}({\mu^\circ}_{s.A.\tau_{\text{suf}}(a)}|P_{\text{pre}}) = \pi^{-1}_{\text{pre}}({\mu^\circ}_s|P_{\text{pre}}) - \hat{\mu}$, $\pi^{-1}_{\text{suf}}({\mu^\circ}_{s.A.\tau_{\text{suf}}(a)}|P_{\text{suf}}) = \pi^{-1}_{\text{suf}}({\mu^\circ}_s|P_{\text{suf}}) + \hat{\mu} - {}^\bullet a + a^\bullet$ donc $\pi^{-1}_{\text{pre}}({\mu^\circ}_{s.A.\tau_{\text{suf}}(a)}|P_{\text{pre}}) + \pi^{-1}_{\text{suf}}({\mu^\circ}_{s.A.\tau_{\text{suf}}(a)}|P_{\text{suf}}) = \pi^{-1}_{\text{pre}}({\mu^\circ}_s|P_{\text{pre}}) + \pi^{-1}_{\text{suf}}({\mu^\circ}_s|P_{\text{suf}}) - {}^\bullet a + a^\bullet = \mu_{u.a} = \mu_{u'}$.

   (c) $\pi^{-1}_{\text{cut}}({\mu^\circ}_{s.A.\tau_{\text{suf}}(a)}|P_{\text{cut}}) = \pi^{-1}_{\text{cut}}({\mu^\circ}_s|P_{\text{cut}}) + \hat{\mu} = \text{req}(w.a)$.

   (d) $\imath \xrightarrow{s} q \xrightarrow{A} q \xrightarrow{\tau_{\text{suf}}(a)} q'$.

2. L'événement e_a apparaît dans $\mathcal{K}_{v'}$. Alors il existe un mot v tel que $\mathcal{K}_{v'} \in [\![v.a]\!]_{\mu_{in}}$ et $\mathcal{K}_u \in [\![v.w']\!]_{\mu_{in}}$. Il s'ensuit que (u, v, w') est une exécution partielle de longueur n. Par hypothèse de récurrence, il existe une séquence de règles tirable s dans $\mathcal{S}^\circ$ telle que les quatre propriétés du lemme sont satisfaites. Comme l'événement e_a se trouve dans le préfixe $\mathcal{K}_{v'}$, l'événement e_a ne consomme pas de jetons produit pas le suffixe $\mathcal{K}_{w'}$. Nous avons alors

$$^\bullet a \leq \mu_v - \mathrm{req}(w') = \pi_{\mathrm{cut}}^{-1}({\mu^\circ}_s|P_{\mathrm{cut}}) + \pi_{\mathrm{pre}}^{-1}({\mu^\circ}_s|P_{\mathrm{pre}}) - \pi_{\mathrm{cut}}^{-1}({\mu^\circ}_s|P_{\mathrm{cut}}) = \pi_{\mathrm{pre}}^{-1}({\mu^\circ}_s|P_{\mathrm{pre}}).$$

Par conséquent $\tau_{\mathrm{pre}}(a)$ peut être exécuté à partir du marquage ${\mu^\circ}_s$ et nous obtenons une nouvelle séquence de règles tirable $s.\tau_{\mathrm{pre}}(a)$. Celle-ci remplit les propriétés requises.

(a) Nous avons $\pi_{\mathrm{cut}}^{-1}({\mu^\circ}_{s.\tau_{\mathrm{pre}}(a)}|P_{\mathrm{cut}}) = \pi_{\mathrm{cut}}^{-1}({\mu^\circ}_s|P_{\mathrm{cut}})$ et $\pi_{\mathrm{pre}}^{-1}({\mu^\circ}_{s.\tau_{\mathrm{pre}}(a)}|P_{\mathrm{pre}}) = \pi_{\mathrm{pre}}^{-1}({\mu^\circ}_s|P_{\mathrm{pre}}) - {}^\bullet a + a^\bullet$, d'où $\pi_{\mathrm{cut}}^{-1}({\mu^\circ}_{s.\tau_{\mathrm{pre}}(a)}|P_{\mathrm{cut}}) + \pi_{\mathrm{pre}}^{-1}({\mu^\circ}_{s.\tau_{\mathrm{pre}}(a)}|P_{\mathrm{pre}}) = \mu_{v.a}$.

(b) Comme $\pi_{\mathrm{suf}}^{-1}({\mu^\circ}_{s.\tau_{\mathrm{pre}}(a)}|P_{\mathrm{suf}}) = \pi_{\mathrm{suf}}^{-1}({\mu^\circ}_s|P_{\mathrm{suf}})$ et $\pi_{\mathrm{pre}}^{-1}({\mu^\circ}_{s.\tau_{\mathrm{pre}}(a)}|P_{\mathrm{pre}}) = \pi_{\mathrm{pre}}^{-1}({\mu^\circ}_s|P_{\mathrm{pre}}) - {}^\bullet a + a^\bullet$ nous obtenons $\pi_{\mathrm{suf}}^{-1}({\mu^\circ}_{s.\tau_{\mathrm{pre}}(a)}|P_{\mathrm{suf}}) + \pi_{\mathrm{pre}}^{-1}({\mu^\circ}_{s.\tau_{\mathrm{pre}}(a)}|P_{\mathrm{pre}}) = \mu_{u.a} = \mu_{u'}$.

(c) $\pi_{\mathrm{cut}}^{-1}({\mu^\circ}_{s.\tau_{\mathrm{pre}}(a)}|P_{\mathrm{cut}}) = \pi_{\mathrm{cut}}^{-1}({\mu^\circ}_s|P_{\mathrm{cut}}) = \mathrm{req}(w')$.

(d) $\imath \xrightarrow{s} q \xrightarrow{\tau_{\mathrm{pre}}(a)} q'$.

□

Lemme 9.2.4. *Soit (u, v, w) une exécution partielle et $a \in R$ une règle telle que $^\bullet a + req(w) \leq \mu_v$. Si $u.a \in CS(\mathcal{S})$ alors $(u.a, v.a, w)$ est une exécution partielle.*

Démonstration. Soit $\mathcal{K}_u \in [\![u]\!]_{\mu_{in}}$ et $\mathcal{K}_v \in [\![v]\!]_{\mu_{in}}$ un préfixe de $\mathcal{K}_u$. Comme w est exécutable depuis $\mathrm{req}(w)$, il existe un réseau causal $\mathcal{K}_w \in [\![w]\!]_{\mathrm{req}(w)}$. Nous pouvons construire un processus $\mathcal{K}'_u$ de u qui admet $\mathcal{K}_v$ comme préfixe et tel que le suffixe $\mathcal{K}'_u \setminus \mathcal{K}_v$ correspond à $\mathcal{K}_w$. En particulier, seulement $\mathrm{req}(w)$ jetons du multi-ensemble μ_v accessible par $\mathcal{K}_v$ sont consommés par $\mathcal{K}_w$. Alors, nous pouvons ajouter une occurrence de la règle a à $\mathcal{K}'_u$ qui consomme $^\bullet a$ jetons de $\mu_v - \mathrm{req}(w)$. □

Inversement, nous devons montrer que le marquage de $P_{\mathrm{pre}} \cup P_{\mathrm{cut}}$ atteint après une séquence de transition tirable s de $\mathcal{S}^\circ$ correspond à un marquage accessible par préfixe. Pour cela, nous devons construire une séquence de règles tirable $u \in \mathrm{FCS}(\mathcal{S})$, un processus $\mathcal{K}_u \in [\![u]\!]_{\mu_{in}}$ et un préfixe $\mathcal{K}_v \in [\![v]\!]_{\mu_{in}}$ inductivement à partir de s. À chaque étape, l'état atteint par s coïncide avec l'état atteint par u. Quand $\mathcal{S}^\circ$ applique un arc supplémentaire a, l'exécution partielle correspondante est soit (u, v, w) si $a \in \xrightarrow{r}_{\mathrm{cut}}$; ou $(u.r, v, w.r)$ si $a \in \xrightarrow{r}_{\mathrm{suf}}$; ou $(u.r, v.r, w)$ si $a \in \xrightarrow{r}_{\mathrm{pre}}$. Dans ce dernier cas, la règle r et la séquence de règles w peuvent être effectuées en même temps : formellement nous allons établir que $^\bullet r + \mathrm{req}(w) \leq \mu_v$. Cette propriété découle en réalité du fait que w peut être exécuté à partir du marquage obtenu par les jetons transférés de P_{pre} vers P_{suf}, c'est-à-dire que $\pi_{\mathrm{cut}}(\mathrm{req}(w)) \leq {\mu^\circ}_s|P_{\mathrm{cut}}$.

Lemme 9.2.5. *Soit s une séquence de règles tirable de $\mathcal{S}^\circ$ conduisant à l'état q et au marquage ${\mu^\circ}_s$. Il existe une exécution partielle (u, v, w) de $\mathcal{S}$ telle que*

1. $\pi_{cut}^{-1}(\mu^\circ{}_s|P_{cut}) + \pi_{pre}^{-1}(\mu^\circ{}_s|P_{pre}) = \mu_v$,
2. $\pi_{suf}^{-1}(\mu^\circ{}_s|P_{suf}) + \pi_{pre}^{-1}(\mu^\circ{}_s|P_{pre}) = \mu_u$,
3. $\pi_{cut}^{-1}(\mu^\circ{}_s|P_{cut}) \geq req(w)$, *et*
4. $\imath \xrightarrow{u} q$ *dans* $\mathcal{S}$.

Démonstration. Nous procédons par récurrence sur la longueur de s. Si $|s| = 0$ alors $s = \varepsilon$; l'exécution partielle vide $(\varepsilon, \varepsilon, \varepsilon)$ satisfait aux quatre conditions, car ${\mu_{in}}^\circ = \pi_{\text{pre}}(\mu_{in})$. L'étape d'induction : soit $s.a$ une séquence de règles tirable de longueur $n+1$. Par hypothèse de récurrence, il existe une exécution partielle (u, v, w) qui remplit les quatre conditions ci-dessus. Nous distinguons trois cas :

1. $q \xrightarrow{a}_{\text{cut}} q$ dans $\mathcal{S}^\circ$. Alors, on peut vérifier que (u, v, w) satisfait aux quatre conditions pour $s.a$.
   (a) $\pi_{\text{cut}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{cut}}) + \pi_{\text{pre}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{pre}}) = \pi_{\text{cut}}^{-1}(\mu^\circ{}_s|P_{\text{cut}}) + \pi_{\text{pre}}^{-1}(\mu^\circ{}_s|P_{\text{pre}}) = \mu_v$.
   (b) $\pi_{\text{suf}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{suf}}) + \pi_{\text{pre}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{pre}}) = \pi_{\text{suf}}^{-1}(\mu^\circ{}_s|P_{\text{suf}}) + \pi_{\text{pre}}^{-1}(\mu^\circ{}_s|P_{\text{pre}}) = \mu_u$.
   (c) Nous avons $\pi_{\text{cut}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{cut}}) > \pi_{\text{cut}}^{-1}(\mu^\circ{}_s|P_{\text{cut}}) \geq \text{req}(w)$.
   (d) $\imath \xrightarrow{u} q$ par hypothèse d'induction.
2. $q \xrightarrow{a}_{\text{suf}} q'$. Alors $\tau_{\text{suf}}^{-1}(a)$ est un arc $q \xrightarrow{r}_{\mathcal{S}} q'$ dans $\mathcal{S}$ et $u.r$ est une séquence de règles de $\mathcal{S}$. De plus, $\pi_{\text{suf}}^{-1}({}^\bullet a|P_{\text{suf}}) = {}^\bullet r$ et $\pi_{\text{suf}}^{-1}(a^\bullet|P_{\text{suf}}) = r^\bullet$. De plus, ${}^\bullet a \leq \mu^\circ{}_s$, donc ${}^\bullet r \leq \pi_{\text{suf}}^{-1}(\mu^\circ{}_s|P_{\text{suf}}) \leq \mu_u$. Par le lemme 9.2.2 nous savons que $(u.r, v, w.r)$ est une exécution partielle de $\mathcal{S}$. De plus,
   (a) $\pi_{\text{cut}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{cut}}) + \pi_{\text{pre}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{pre}}) = \pi_{\text{cut}}^{-1}(\mu^\circ{}_s|P_{\text{cut}}) + \pi_{\text{pre}}^{-1}(\mu^\circ{}_s|P_{\text{pre}}) = \mu_v$.
   (b) $\pi_{\text{suf}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{suf}}) + \pi_{\text{pre}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{pre}}) = \pi_{\text{suf}}^{-1}(\mu^\circ{}_s|P_{\text{suf}}) + \pi_{\text{pre}}^{-1}(\mu^\circ{}_s|P_{\text{pre}}) + \pi_{\text{suf}}^{-1}(a^\bullet - {}^\bullet a|P_{\text{suf}}) = \mu_u - {}^\bullet r + r^\bullet = \mu_{u.r}$.
   (c) Nous avons $\mu_{s.a}|P_{\text{cut}} = \mu_s|P_{\text{cut}}$. De plus, ${}^\bullet a \leq \mu^\circ{}_s$, ainsi

   $$\pi_{\text{cut}}^{-1}(\mu^\circ{}_s|P_{\text{cut}}) \geq \pi_{\text{cut}}^{-1}(\mu^\circ{}_s|P_{\text{cut}}) + \pi_{\text{suf}}^{-1}({}^\bullet a - \mu^\circ{}_s|P_{\text{suf}}) = \pi_{\text{suf}}^{-1}({}^\bullet a|P_{\text{suf}}) + (\mu_v - \mu_u) = {}^\bullet r + \text{cost}(w)$$

   Alors, $\pi_{\text{cut}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{cut}}) = \pi_{\text{cut}}^{-1}(\mu^\circ{}_s|P_{\text{cut}}) \geq \max(\text{req}(w), {}^\bullet r + \text{cost}(w)) = \text{req}(w.r)$.
   (d) $\imath \xrightarrow{u} q \xrightarrow{r} q'$ et $\imath \xrightarrow{s} q \xrightarrow{a} q'$.
3. $q \xrightarrow{a}_{\text{pre}} q'$. Alors $\tau_{\text{pre}}^{-1}(a)$ est un arc $q \xrightarrow{r}_{\mathcal{S}} q'$. De plus, $\pi_{\text{pre}}^{-1}({}^\bullet a|P_{\text{pre}}) = {}^\bullet r$ et $\pi_{\text{pre}}^{-1}(a^\bullet|P_{\text{pre}}) = r^\bullet$. Nous observons tout d'abord que
   (a) $\pi_{\text{cut}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{cut}}) + \pi_{\text{pre}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{pre}}) = \mu_v - \pi_{\text{pre}}^{-1}({}^\bullet a|P_{\text{pre}}) + \pi_{\text{pre}}^{-1}(a^\bullet|P_{\text{pre}}) = \mu_{v.r}$
   (b) $\pi_{\text{suf}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{suf}}) + \pi_{\text{pre}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{pre}}) = \mu_u - \pi_{\text{pre}}^{-1}({}^\bullet a|P_{\text{pre}}) + \pi_{\text{pre}}^{-1}(a^\bullet|P_{\text{pre}}) = \mu_{u.r}$
   (c) $\pi_{\text{cut}}^{-1}(\mu^\circ{}_{s.a}|P_{\text{cut}}) = \pi_{\text{cut}}^{-1}(\mu^\circ{}_s|P_{\text{cut}}) \geq \text{req}(w)$
   (d) $q \xrightarrow{r} q'$ dans $\mathcal{S}$.

   Puisque $\mu^\circ{}_s \geq {}^\bullet a$, nous avons $\pi_{\text{pre}}^{-1}(\mu^\circ{}_s|P_{\text{pre}}) \geq \pi_{\text{pre}}^{-1}({}^\bullet a|P_{\text{pre}})$. En revanche, $\pi_{\text{cut}}^{-1}(\mu^\circ{}_s|P_{\text{cut}}) \geq \text{req}(w)$, ainsi $\pi_{\text{pre}}^{-1}(\mu^\circ{}_s|P_{\text{pre}}) + \pi_{\text{cut}}^{-1}(\mu^\circ{}_s|P_{\text{cut}}) \geq \pi_{\text{pre}}^{-1}({}^\bullet a|P_{\text{pre}}) + \text{req}(w)$, c'est-à-dire que $\mu_v \geq {}^\bullet r + \text{req}(w)$. Par le lemme 9.2.4, $(u.r, v.r, w)$ est une exécution partielle.

□

Nous sommes maintenant prêts à prouver la propriété 9.1.1. Soit μ le marquage atteint par un préfixe $\mathcal{K}'$ d'un processus $\mathcal{K} \in [\![\mathcal{S}]\!]$. Selon la propriété 9.2.1, il existe une exécution partielle (u, v, w) telle que $\mu_v = \mu$. Par le lemme 9.2.3, il existe une séquence de règles tirable s dans $\mathcal{S}^\circ$ telle que $\pi_{\text{cut}}^{-1}(\mu^\circ s|P_{\text{cut}}) + \pi_{\text{pre}}^{-1}(\mu^\circ s|P_{\text{pre}}) = \mu_v = \mu$. Inversement, si $\pi_{\text{cut}}^{-1}(\mu^\circ s|P_{\text{cut}}) + \pi_{\text{pre}}^{-1}(\mu^\circ s|P_{\text{pre}}) = \mu$ pour une certaine séquence de règles tirable s dans $\mathcal{S}^\circ$ alors le lemme 9.2.5 assure qu'il existe une exécution partielle (u, v, w) telle que $\pi_{\text{cut}}^{-1}(\mu^\circ s|P_{\text{cut}}) + \pi_{\text{pre}}^{-1}(\mu^\circ s|P_{\text{pre}}) = \mu_v$. De plus, la propriété 9.2.1 affirme que μ_v est le marquage atteint par un préfixe $\mathcal{K}'$ d'un processus $\mathcal{K} \in [\![\mathcal{S}]\!]$.

9.3 Analyse des marquages accessibles par préfixe

La propriété 9.1.1 nous permet d'utiliser des techniques existantes pour les étendre à l'analyse des marquages accessibles par préfixe de $\mathcal{S}$. Tout d'abord, le problème du *caractère borné des préfixes* demande si l'ensemble des marquages accessibles par préfixe d'un PNS donné $\mathcal{S}$ est fini. Il est facile de prouver que le PNS $\mathcal{S}$ est borné par préfixe si, et seulement si, le PNS $\mathcal{S}^\circ$ est borné. De plus, ceci peut être vérifié au moyen de la simulation linéaire habituelle d'un PNS par un réseau de Petri. Par ailleurs, la borne par préfixe est équivalente à la borne dans le cas particulier des réseaux de Petri, car l'ensemble des processus d'un réseau de Petri est clos par préfixes. Ainsi,

Théorème 9.3.1. *Le problème du caractère borné par préfixe d'un PNS est équivalent au problème du caractère borné d'un réseau de Petri.*

Démonstration. Il est clair, à partir de la propriété 9.1.1, que si $\mathcal{S}^\circ$ est borné alors il y a un nombre fini de marquages accessibles par préfixe dans $\mathcal{S}$. Réciproquement, supposons que $\mathcal{S}$ est borné par préfixe. Alors il existe $M \in \mathbb{N}$ tel que $\mu_v(p) \leq M$ et $\mu_u(p) \leq M$ pour toutes exécutions partielles (u, v, w) de $\mathcal{S}$ et toutes places $p \in P$. Alors le lemme 9.2.5 garantit que si une séquence de règles tirable de $\mathcal{S}^\circ$ conduit à un marquage μ, alors $\mu(p) \leq M$ pour chaque place $p \in P_{\mathcal{S}^\circ}$. □

Deuxièmement, le problème de la *couverture par préfixe* demande si un multi-ensemble de places $\mu \in \mathbb{N}^P$ donné est couvert par le marquage accessible par préfixe $\mu' \in \mathbb{N}^P$, c'est-à-dire que $\mu(p) \leq \mu'(p)$ pour tout $p \in P$. Il est facile de voir que μ est couvert par préfixe dans $\mathcal{S}$ si, et seulement si, le multi-ensemble de places $\pi_{\text{cut}}(\mu)$ est couvert par un marquage accessible $\mathcal{S}^\circ$. Ainsi,

Théorème 9.3.2. *Le problème de la couverture par préfixe d'un PNS est équivalent au problème de la couverture d'un réseau de Petri.*

Démonstration. Le problème de la couverture par préfixe d'un réseau de Petri est équivalent au problème de couverture des réseaux de Petri. Ainsi, le problème de la couverture par préfixe d'un PNS est au moins aussi difficile que le problème de la couverture des réseaux de Petri. En outre, la vérification de la couverture d'un marquage dans un PNS peut être réalisée au moyen de la simulation linéaire d'un PNS par un réseau de Petri. Il nous faut

donc simplement montrer qu'un marquage μ_0 est couvert par préfixe dans $\mathcal{S}$ si, et seulement si, le multi-ensemble de places $\pi_{\text{cut}}(\mu_0)$ est couvert par un marquage accessible de $\mathcal{S}^\circ$.

Soit $\mu_0 \in \mathbb{N}^P$ un multi-ensemble de places. Supposons d'abord que μ_0 est couvert par un marquage accessible par préfixe μ de $\mathcal{S}$: $\mu_0 \leq \mu$. Par la propriété 9.1.1, il existe un marquage accessible μ° dans $\mathcal{S}^\circ$ tel que $\mu = \pi_{\text{pre}}^{-1}(\mu^\circ|P_{\text{pre}}) + \pi_{\text{cut}}^{-1}(\mu^\circ|P_{\text{cut}})$. En raison des boucles de $\longrightarrow_{\text{cut}}$ dans $\mathcal{S}^\circ$, on peut supposer que $\mu^\circ|P_{\text{pre}} = 0$ ainsi $\pi_{\text{cut}}(\mu) = \mu^\circ|P_{\text{cut}}$ et $\pi_{\text{cut}}(\mu_0) \leq \mu^\circ|P_{\text{cut}}$.

Réciproquement, supposons maintenant que $\pi_{\text{cut}}(\mu_0) \leq \mu^\circ$ pour un marquage accessible μ° de $\mathcal{S}^\circ$. Alors, par le lemme 9.2.5 et la propriété 9.2.1, $\mu_0 \leq \pi_{\text{cut}}^{-1}(\mu^\circ|P_{\text{cut}}) \leq \mu$ pour un marquage préfixe accessible μ de $\mathcal{S}$. □

Pour terminer, le problème de l'*accessibilité par préfixe* demande si un multi-ensemble de places donné est accessible par préfixe dans $\mathcal{S}$. Considérons une légère modification $\mathcal{S}'$ de $\mathcal{S}^\circ$ où, pour chaque place $p \in P_{\text{suf}}$, chaque état q de $\mathcal{S}^\circ$ est muni d'un arc boucle supplémentaire qui porte une règle qui consomme un jeton de p et ne produit rien. Alors, un multi-ensemble μ de places est accessible par préfixe dans $\mathcal{S}$ si, et seulement si, $\pi_{\text{cut}}(\mu)$ est accessible dans $\mathcal{S}'$. Ainsi,

Théorème 9.3.3. *Le problème de l'accessibilité par préfixe d'un PNS est équivalent au problème de l'accessibilité dans un réseau de Petri.*

Démonstration. Le problème de l'accessibilité par préfixe d'un réseau de Petri est équivalent au problème de l'accessibilité d'un réseau de Petri. Ainsi, le problème de l'accessibilité par préfixe d'un PNS est au moins aussi difficile que le problème de l'accessibilité des réseaux de Petri. De plus, la vérification de l'accessibilité d'un marquage dans un PNS peut être réalisée au moyen de la simulation linéaire d'un PNS par un réseau de Petri. Ainsi, nous avons simplement besoin de montrer qu'un multi-ensemble μ de places est accessible par préfixe dans $\mathcal{S}$ si, et seulement si, $\pi_{\text{cut}}(\mu)$ est accessible dans $\mathcal{S}'$.

Soit $\mu \in \mathbb{N}^P$ un multi-ensemble de places. Supposons d'abord que μ est accessible par préfixe dans $\mathcal{S}$. Par la propriété 9.1.1, il existe un marquage accessible μ° dans $\mathcal{S}^\circ$ tel que $\mu = \pi_{\text{pre}}^{-1}(\mu^\circ|P_{\text{pre}}) + \pi_{\text{cut}}^{-1}(\mu^\circ|P_{\text{cut}})$. En raison des boucles $\longrightarrow_{\text{cut}}$ dans $\mathcal{S}^\circ$, on peut supposer que $\mu^\circ|P_{\text{pre}} = 0$ et donc $\pi_{\text{cut}}(\mu) = \mu^\circ|P_{\text{cut}}$. Alors $\pi_{\text{cut}}(\mu)$ est accessible dans $\mathcal{S}'$ car les boucles supplémentaires de $\mathcal{S}'$ nous permettent de retirer tous les jetons dans toutes les places de P_{suf}.

Réciproquement, supposons que $\pi_{\text{cut}}(\mu)$ est accessible dans $\mathcal{S}'$. Alors il existe un marquage accessible μ° dans $\mathcal{S}^\circ$ tel que $\pi_{\text{cut}}(\mu) = \mu'|P_{\text{cut}}$ et $\mu^\circ|P_{\text{pre}} = 0$. Il résulte du lemme 9.2.5 et de la propriété 9.2.1 que μ est accessible par préfixe dans $\mathcal{S}$. □

9.4 Analyse des marquages accessibles par préfixe sous la restriction FIFO

Dans la section 7.3, nous avons abordé la modélisation de MSG par des PNS. Nous avons défini la classe des MSG étendus (définition 7.3.3) qui impose certaines restrictions aux PNS

considérés. Comme les MSG sont généralement utilisés sous l'hypothèse de communications FIFO, nous avons également introduit cette notion pour les MSG étendus (définition 7.3.4). Nous montrons dans cette section comment les résultats obtenus dans ce chapitre s'appliquent lorsque nous adoptons la restriction FIFO sur les MSG étendus.

Afin de simplifier les preuves, on note par $[\![u]\!]_{\mu_{in}}$ la classe de tous les processus FIFO de u à partir du marquage μ_{in}.

Propriété 9.4.1. *Soit $\mathcal{S}$ un MSG étendu. Si (u, v, w) est une exécution partielle non vide, alors $[\![u]\!]_{\mu_{in}} \cap [\![v.w]\!]_{\mu_{in}} \neq \emptyset$.*

Démonstration. Une règle $r = (\lambda, \alpha, \beta)$ est dite localisée si elle utilise une instance. C'est-à-dire qu'il existe une instance i tel que $i \in \alpha$.

Nous procédons par induction sur la longueur de u. Si $|u| = 1$, alors $u = r$, $v = \epsilon$ et $w = r$ ou $u = r$, $v = r$ et $w = \epsilon$. Comme $[\![r]\!]_{\mu_{in}} = [\![r.\epsilon]\!]_{\mu_{in}} = [\![\epsilon.r]\!]_{\mu_{in}}$, la propriété est vérifiée.

Étape d'induction : nous considérons l'exécution partielle (u', v', w') avec $|u'| = n + 1$. Soit $u' = u.r$ avec $|u| = n$. Puisque (u', v', w') est une exécution partielle, on a $[\![u']\!]_{\mu_{in}} \cap [\![v'.w']\!]_{\mu_{in}} \neq \emptyset$ d'après la propriété 9.2.1.

Cas 1 La règle r est localisée. Soit u_1 et u_2 tels que $u_1.r.u_2 = v'.w'$ avec r n'apparaissant pas dans u_2. Comme r est localisée, cette règle consomme un jeton d'une instance i pour le restituer. Comme les jetons des instances sont uniques, toutes les règles localisées sur l'instance i sont linéairement ordonnées. Ainsi, le dernier événement de la règle r dans un processus $\mathcal{K}'$ de $[\![u']\!]_{\mu_{in}} \cap [\![v'.w']\!]_{\mu_{in}}$ correspond au dernier événement localisé sur i dans $\mathcal{K}'$. Donc u_2 ne contient pas de règle localisée sur l'instance i. Considérons le processus $\mathcal{K}$ préfixe de $\mathcal{K}'$ obtenu en retirant la dernière occurrence de r. Alors $\mathcal{K} \in [\![u]\!]_{\mu_{in}}$ et $\mathcal{K} \in [\![u_1.u_2]\!]_{\mu_{in}}$. Ainsi, $[\![u]\!]_{\mu_{in}} \cap [\![u_1.u_2]\!]_{\mu_{in}} \neq \emptyset$ et donc (u, u_1, u_2) est une exécution partielle, ce qui implique par hypothèse d'induction $[\![u]\!]_{\mu_{in}} \cap [\![u_1.u_2]\!]_{\mu_{in}} \neq \emptyset$. On peut ajouter la règle r à un tel processus FIFO pour obtenir un nouveau processus FIFO. Ainsi, $[\![u.r]\!]_{\mu_{in}} \cap [\![u_1.u_2.r]\!]_{\mu_{in}} \neq \emptyset$. Comme r est localisée sur l'instance i et que u_2 ne contient pas de règle localisée sur i, on a $[\![u_1.r.u_2]\!]_{\mu_{in}} = [\![u_1.u_2.r]\!]_{\mu_{in}}$. Ainsi, $[\![u_1.r.u_2]\!]_{\mu_{in}} \cap [\![u']\!]_{\mu_{in}} \neq \emptyset$.

Cas 2 La règle r n'est pas localisée. Si u' ne contient pas de règle localisée, alors $[\![u']\!]_{\mu_{in}} = [\![u']\!]_{\mu_{in}}$ donc $[\![v'.w']\!]_{\mu_{in}} = [\![v'.w']\!]_{\mu_{in}}$ et donc $[\![u']\!]_{\mu_{in}} \cap [\![v'.w']\!]_{\mu_{in}} \neq \emptyset$.
Si u' contient une règle localisée, alors nous considérons $u' = u_1.b.u_2.r$ avec b une règle localisée, et u_2 une séquence de règles ne contenant pas de règle localisée. Comme $u_2.r$ ne dépend pas de b, $[\![u_1.b.u_2.r]\!]_{\mu_{in}} = [\![u_1.u_2.r.b]\!]_{\mu_{in}}$ i.e. $[\![u']\!]_{\mu_{in}} = [\![u_1.u_2.r.b]\!]_{\mu_{in}}$. Par conséquent, $[\![u']\!]_{\mu_{in}} = [\![u_1.u_2.r.b]\!]_{\mu_{in}}$. Notons $u'' = u_1.u_2.r.b$. Alors $[\![u'']\!]_{\mu_{in}} \cap [\![v'.w']\!]_{\mu_{in}} = [\![u']\!]_{\mu_{in}} \cap [\![v'.w']\!]_{\mu_{in}} \neq \emptyset$ et donc (u'', v', w') est une exécution partielle. D'après le premier cas, $[\![u'']\!]_{\mu_{in}} \cap [\![v'.w']\!]_{\mu_{in}} \neq \emptyset$, i.e. $[\![u']\!]_{\mu_{in}} \cap [\![v'.w']\!]_{\mu_{in}} \neq \emptyset$.

□

Étant donné que s'il existe un marquage accessible par préfixe dans un MSG étendu, alors ce même marquage est accessible sous la restriction FIFO, tous les résultats vus précédemment, tels que les théorèmes 9.3.1, 9.3.2 et 9.3.3, fonctionnent également lorsque nous

adoptons la restriction FIFO pour les MSG étendus. En particulier, nous pouvons vérifier la couverture ou l'accessibilité par préfixes d'un marquage donné pour n'importe quel MSG étendus (éventuellement non borné par préfixe).

9.5 Résultats expérimentaux

La notion de non-divergence (aussi appelée borne universellement) pour un MSG coïncide avec le caractère borné par préfixe pour les PNS. Cependant, l'expressivité des PNS est plus grande que celle des MSG. Alors que la non-divergence d'un MSG est coNP-complet, le caractère borné par préfixe d'un PNS requiert un espace exponentiel. Bien que les deux problèmes sont dans des classes de complexité distinctes, nous comparons à titre indicatif nos algorithmes à l'outil vérifiant la divergence de MSG conçu dans la partie I.

Nous avons implémenté notre construction dans un prototype (baptisé VaChe [5]) afin de vérifier le caractère borné par préfixe en utilisant TINA [4]. Nous présentons ici les résultats que nous obtenons sur la modélisation d'une version simplifiée du protocole des fenêtres coulissantes, illustrée sur la figure 9.2. Cette modélisation contient deux états. Le premier état envoie un message de i vers j, le second état envoie un message de j vers i. Chaque fois que l'on accède au premier état, la valeur du compteur c est incrémentée. Lorsque l'on accède au second état, la valeur du compteur c est décrémentée. Initialement, $c = 1$. Notons qu'à tout instant $0 \leq c \leq 4$. Ainsi, ce protocole permet au serveur i d'envoyer des messages à un client j avec un maximum de 4 acquittements manquants, c'est-à-dire 4 messages dans la fenêtre.

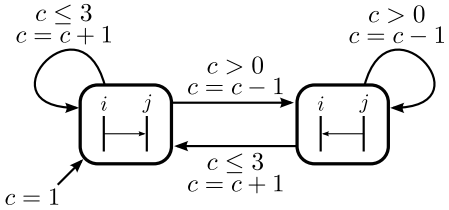


FIGURE 9.2 – Modélisation du protocole simplifié des fenêtres coulissantes avec un PNS.

Nous avons vérifié le caractère préfixe borné de ce PNS en faisant varier la taille de la fenêtre. Les résultats obtenus sont présentés sur la figure 9.3.

n (taille de fenêtre)	VaChe (préfixe borné)	τ_{DIV}
40	9	<0.01
60	37	<0.01
80	110	<0.01
100	256	<0.01

FIGURE 9.3 – Temps d'exécution en sec. pour le protocole simplifié des fenêtres coulissantes.

Le PNS de la figure 9.2 est équivalent au MSG de la figure 4.1(a). Comme nous l'avions déjà remarqué, le PNS est non seulement plus concis que le MSG, mais il est également

très simple à comprendre. En effet, le PNS contient deux nœuds. Le premier nœud envoie un message de i vers j et le second nœud envoie un acquittement de j vers i. Enfin, une variable c est utilisée pour restreindre le nombre d'acquittements manquants.

Nous avons cependant constaté sur cet exemple qu'il est beaucoup plus efficace de vérifier la divergence d'un MSG, en utilisant les techniques développées dans la partie I, que de vérifier la divergence du PNS correspondant. Cela n'est pas surprenant, car les PNS sont plus généraux que les MSG. Afin d'améliorer la vérification de PNS, nous allons nous intéresser dans la dernière partie aux propriétés structurelles et semi-structurelles. L'idée consiste à ne pas fixer le marquage initial (ou une partie du marquage initial) afin de diminuer la difficulté du problème.

Troisième partie

Propriétés structurelles

Chapitre

10 Introduction

Comme nous l'avons expliqué dans la section 7.3, les PNS peuvent être utilisés afin de modéliser des MSG étendus. Ces MSG étendus apportent de nombreux avantages par rapport aux MSG classiques, notamment au niveau de l'expressivité et de la concision. Rappelons en effet que nous pouvons représenter de manière exponentiellement plus concise un même protocole de communication comme dans l'exemple du protocole simplifié des fenêtres coulissantes (figure 7.6).

Un problème classique sur les MSG est le problème de la divergence (définition 4.3.2). Ce problème consiste à vérifier s'il y a un nombre non borné de messages dans les canaux de communication. Alors que ce problème est NP-complet [11, Th. 7], le problème équivalent dans le cadre plus général des PNS nécessite un espace exponentiel (propriété 9.3.1). D'une certaine manière, cette différence de complexité entre les deux modèles n'est pas surprenante, car d'une part les MSG étendus sont plus expressifs et d'autre part les MSG étendus permettent des modélisations exponentiellement plus concises. Il existe cependant des protocoles pouvant être modélisés par des MSG qui font la même taille qu'une modélisation avec des MSG étendus. Dans ce cas particulier, la différence de complexité entre les deux modèles peut être gênante.

Pour contourner cette difficulté pratique, nous proposons d'abstraire les propriétés à vérifier, de sorte que si l'abstraction d'une propriété est satisfaite, alors cette propriété est satisfaite. L'abstraction que nous proposons consiste à vérifier les propriétés de manière structurelle, c'est-à-dire à vérifier si, quel que soit le marquage initial, la propriété est satisfaite. Nous pouvons citer comme exemple le caractère structurellement borné qui assure que, quel que soit le marquage initial d'un réseau de Petri, l'ensemble des marquages accessibles est fini. Cette abstraction est intéressante, car vérifier le caractère borné d'un réseau de Petri fourni avec une configuration initiale nécessite un espace exponentiel [43, 74] alors que vérifier le caractère structurellement borné est polynomial [43, 81].

Il est aussi intéressant de pouvoir définir le degré d'abstraction souhaité. En effet, si une propriété n'est pas vraie structurellement, nous ne pouvons rien dire sur la véracité de cette propriété pour un marquage initial donné. Ceci est d'autant plus vrai pour les MSG étendus. Une instance I étant vue comme un jeton dans une place I, la vérification de propriétés structurelles abstrairait également les processus en ne considérant pas que chaque instance est unique. Cette abstraction est souvent trop forte et beaucoup de propriétés vraies s'avèrent fausses structurellement.

Afin de pouvoir contrôler le degré d'abstraction, nous introduisons la notion de propriété semi-structurelle. Cela consiste à fixer le marquage initial d'un sous-ensemble approprié de places, puis à vérifier les propriétés structurelles sur les places restantes. Pour cela, nous déplions le PNS afin de faire disparaître les places dont nous fixons le marquage initial. Les propriétés structurelles sur le PNS ainsi obtenu impliquent les propriétés semi-structurelles du PNS d'origine. Notons que si nous fixons le marquage initial pour la totalité des places

et que le PNS est borné, alors le dépliage correspond à un PNS de taille exponentielle relativement à la borne. Si par contre aucune place n'est fixée, alors aucun dépliage n'est effectué et la vérification se fait en temps polynomial. Nous pouvons ainsi jouer sur les places dont nous fixons le marquage initial afin de vérifier une propriété.

Vérifier le caractère structurellement borné ou la terminaison structurelle d'un PNS se ramène à vérifier les coûts des cycles : un cycle est pathologique pour le caractère structurellement borné (respectivement la terminaison structurelle) si la somme des étiquettes du cycle est strictement positive (respectivement positive). Ainsi, nous pouvons considérer les PNS comme des VASS en remplaçant chaque règle par leur vecteur bilan. Par conséquent, ces deux problèmes sont très proches de celui de la détection d'un *zéro-cycle* dans les graphes dynamiques [64], qui recherche dans un VASS l'existence d'un cycle avec un coût nul. Dans [68], Kosaraju et Sullivan ont montré comment vérifier l'existence d'un tel cycle en temps polynomial. Par la suite, l'équivalence de ce problème avec celui de la programmation linéaire générale a été prouvée [32]. L'idée est double. Premièrement, les cycles sont identifiés par des multi-ensembles particuliers d'arcs. Deuxièmement, les multi-ensembles d'arcs avec un coût nul apparaissent comme des solutions de certains programmes linéaires. Cette technique s'adapte facilement à la détection de cycles pathologiques pour le problème du caractère structurellement borné et de la terminaison structurelle. L'algorithme retourne, en temps polynomial, un multi-ensemble d'arcs qui représente un cycle pathologique si un tel cycle existe.

Une caractéristique particulièrement attrayante des PNS réside dans leur représentation graphique similaire à un automate. Il est donc intéressant de décrire les bugs de manière visuelle. Toutefois, la longueur minimale d'un cycle pathologique peut être exponentielle en la taille du PNS. Par conséquent, l'énumération de la séquence des arcs décrivant un bug est prohibitive et nous avons besoin de concevoir une *représentation* compacte des cycles pathologiques. Nous proposons de décrire les bugs structurels sous la forme d'un multi-ensemble de cycles particulièrement simples appelés *lassos*. Schématiquement, un lasso comprend un cycle muni de deux chemins aller et retour vers un nœud initial fixé. On exige que la longueur des trois trajets constitutifs du lasso soit au plus égale au nombre de nœuds du VASS. Dans les faits, nous considérons souvent des lassos élémentaires, c'est-à-dire des lassos dont chaque composante est un chemin élémentaire. En outre, la *valuation* d'un lasso détermine le nombre d'itérations de sa composante cyclique. Notre premier résultat montre comment calculer en temps polynomial un multi-ensemble de lassos élémentaires avec un nœud de départ commun qui correspond à un multi-ensemble d'arcs représentant un cycle pathologique. Il est intéressant de noter que le nombre de lassos élémentaires distincts dont nous avons besoin est au plus égal à la dimension des vecteurs de poids des étiquettes du VASS. De plus, le point de départ commun de ces lassos peut être choisi arbitrairement.

Trouver un contre-exemple d'une propriété le plus court possible est souvent souhaitable en pratique pour comprendre le mauvais comportement d'un système [30, 69]. Le nombre d'arcs (ou simplement le nombre d'arcs distincts) dans un cycle pathologique est une mesure intéressante de la complexité d'un bug structurel. Cependant la recherche d'un cycle pathologique avec une longueur minimale (ou un nombre minimal d'arcs distincts) est un

problème NP-complet. Par la suite, nous nous fixerons un nœud de départ $\hat{q}$ et un entier naturel n et nous nous concentrerons sur des lassos partant de $\hat{q}$ avec une longueur d'au plus n. Au moyen d'un codage en programmation linéaire et d'un algorithme de séparation, nous montrerons comment vérifier s'il existe un multi-ensemble pathologique formé par ces lassos, et si oui, comment en calculer un. Par conséquent, nous pourrons calculer en temps polynomial un multi-ensemble pathologique de lassos de longueur minimale.

10.1 Propriétés structurelles et caractérisation par des cycles

Un VASS muni d'une configuration initiale (q_{in}, r_{in}) est *terminant* s'il existe un entier naturel k tel que la longueur de chaque exécution de $\mathcal{S}$ à partir de (q_{in}, r_{in}) est inférieure à k. Autrement dit, un VASS termine si, et seulement si, il n'a pas d'exécution infinie. Nous nous intéressons ici aux propriétés structurelles qui, elles, ne dépendent pas de la configuration initiale.

Définition 10.1.1. *Un VASS $\mathcal{S}$ termine structurellement s'il termine pour toute configuration initiale.*

Le problème de la terminaison structurelle consiste à savoir si un VASS donné n'a pas d'exécution infinie, et ceci quelle que soit la configuration initiale. On constate que cette question se résume à chercher un cycle particulier dans le VASS. Nous notons $cout(\gamma)$ le *coût* du cycle γ correspondant a la somme des vecteur constituant γ.

Proposition 10.1.2. *Un VASS $\mathcal{S}$ termine structurellement si, et seulement si, il n'existe pas de cycle γ avec $cout(\gamma) \geq \vec{0}$.*

Démonstration. Supposons qu'il existe un cycle γ avec $cout(\gamma) \geq \vec{0}$. Alors $\mathcal{S}$ ne termine pas pour certaines configurations initiales. Inversement, si $\mathcal{S}$ ne termine pas pour certaines configurations initiales, alors le dépliage de toutes les exécutions sous la forme d'un arbre est infini. L'arbre correspondant au dépliage des exécutions est généralement appelé *arbre d'accessibilité*. Comme chaque nœud de l'arbre d'accessibilité à au plus $|A|$ successeur, le lemme de Kőnig [65] assure que cet arbre contient une branche infinie $s_1, s_2, ...$ Cela donne une suite infinie $(q_1, r_1), (q_2, r_2), ...$ de configurations atteintes le long de cette branche. Rappelons qu'un *bel ordre partiel* est un ensemble partiellement ordonné dans lequel chaque suite infinie contient une sous-suite infinie monotone non décroissante. De plus, le lemme de Dickson [36, Lemma A] montre que $(\mathbb{N}^p, \leq)$ est un bel ordre partiel. Ainsi, il y a des indices $i_1 < i_2 < ...$ tels que la sous-suite infinie $r_{i_1}, r_{i_2}, ...$ satisfaite $r_{i_1} \leq r_{i_2} \leq ....$ Considérons deux indices $i_k < i_l$ tels que $q_{i_k} = q_{i_l}$. Alors la suite de configuration $(q_{i_k}, r_{i_k})(q_{i_k+1}, r_{i_k+1})...(q_{i_l}, r_{i_l})$ correspond à un cycle γ de $\mathcal{S}$ tel que $cout(\gamma) = r_{i_l} - r_{i_k} \geq \vec{0}$ □

Exemple 10.1.3. *Considérons un VASS deux dimensionnel avec trois états q_0, q_1, q_2 et cinq arcs pondérés a_1, a_2, a_3, l_1 et l_2 représentés sur la figure 10.1. Le coût du cycle $\gamma = a_1.l_1^5.a_2.l_2^3.a_3$ vaut $cout(\gamma) = (1, 4)$. Ainsi, ce VASS n'est pas structurellement terminant.*

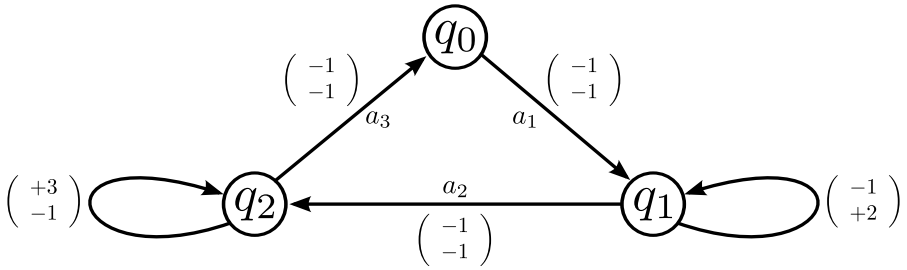


FIGURE 10.1 – Un Vector Addition System with State (VASS).

Un VASS muni d'une configuration initiale (q_{in}, r_{in}) est *borné* s'il existe un entier naturel k tel que, pour chaque configuration (q, r) accessible à partir de (q_{in}, r_{in}), il n'y a jamais plus de k jetons de chaque type, c'est-à-dire $r[i] \leq k$ pour chaque $i \in [i..p]$.

Définition 10.1.4. *Un VASS $\mathcal{S}$ est* structurellement borné *s'il est borné pour toute configuration initiale.*

Similairement à la propriété 10.1.2, vérifier si un VASS est structurellement borné consiste à détecter un cycle avec un coût strictement positif.

Proposition 10.1.5. *Un VASS $\mathcal{S}$ est structurellement borné si, et seulement si, il n'existe pas de cycle γ avec $cout(\gamma) \gneq \vec{0}$*

Démonstration. Supposons qu'il existe un cycle γ avec $cout(\gamma) \gneq \vec{0}$. Alors $\mathcal{S}$ n'est pas borné pour certaines configurations initiales. Inversement, si $\mathcal{S}$ n'est pas borné pour certaines configurations initiales, alors nous considérons le sous arbre de l'arbre d'accessibilité tel que chaque paire de nœuds distincts le long d'une branche correspond à des configurations distinctes. Ce sous-arbre est toujours infini, car l'ensemble des configurations accessibles reste le même. Par le lemme de Kőnig, le sous-arbre d'accessibilité contient une branche infinie $s_1, s_2, ...$ Cela donne une suite infinie $(q_1, r_1), (q_2, r_2), ...$ de configuration atteinte le long de cette branche. Par le lemme de Dickson, il existe des indices $i_1 < i_2 < ...$ tels que la sous-suite infinie $r_{i_1}, r_{i_2}, ...$ satisfaite $r_{i_1} \leq r_{i_2} \leq ....$ Comme $\mathcal{S}$ a un nombre fini d'états, on peut supposer que $q_{i_j} = q_{i_k}$ pour certains j, k. Comme toutes les configurations le long d'une branche sont distinctes, nous obtenons $r_{i_1} \lneq r_{i_2} \lneq r_{i_3} \lneq ....$ Nous pouvons donc trouver, le long de cette branche, un cycle γ avec $cout(\gamma) \gneq \vec{0}$. □

Ainsi, vérifier le caractère structurellement borné et la terminaison structurelle d'un VASS se ramène à détecter des cycles pathologiques dans $\mathcal{S}$.

Définition 10.1.6. *Un cycle γ dans un VASS $\mathcal{S}$ est pathologique pour la terminaison structurelle (respectivement le caractère structurellement borné) si $cout(\gamma) \geq \vec{0}$ (respectivement $cout(\gamma) \gneq \vec{0}$).*

10.2 Cas particulier des réseaux de Petri

Pour le cas particulier des réseaux de Petri, le caractère structurellement borné est appelé *strong boundedness* dans [81] et a été caractérisé par un système linéaire homogène

d'inéquations diophantiennes [81, Th. 1]. Cette technique a été généralisée pour la terminaison structurelle dans [100, Th. 2]. Ces deux résultats peuvent être reformulés dans le cadre des VASS de la manière suivante.

Théorème 10.2.1. *Soit $\mathcal{S}$ un VASS à un seul état. Soit C une matrice d'entier $p \times |A|$ construite avec les $|A|$ vecteurs colonnes $\{cout(a)|a \in A\}$. Alors*

1. *$\mathcal{S}$ est structurellement borné si, et seulement si, il n'existe pas de vecteur $x \in \mathbb{N}^A$ tel que $Cx \gneq \vec{0}$.*
2. *$\mathcal{S}$ termine structurellement si, et seulement si, il n'existe pas de vecteur non nul $x \in \mathbb{N}^A$ tel que $Cx \geq \vec{0}$.*

Ici, tout multi-ensemble $x \in \mathbb{N}^A \setminus \{\vec{0}\}$ représente un cycle γ dans $\mathcal{S}$ pour lequel chaque arc $a \in A$ apparaît exactement $x(a)$ fois, et donc $cout(\gamma) = Cx$. De plus, il est clair que la recherche de solutions à ces deux systèmes linéaires homogènes d'inéquations diophantiennes peut être effectuée avec des nombres rationnels et donc en temps polynomial. Cependant, le nombre minimal d'arcs d'un multi-ensemble pathologique peut être exponentiel dans la taille du réseau de Petri comme nous pouvons le voir dans l'exemple suivant.

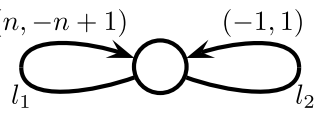


FIGURE 10.2 – Un Vector Addition System with State (VASS).

Exemple 10.2.2. *Considérons le VASS deux dimensionnel $\mathcal{S}$ avec un seul état de la figure 10.2. Il est clair que la longueur minimale d'un cycle γ avec $cout(\gamma) > 0$ est exactement de n avec par exemple $\gamma = l_1.l_2^{n-1}$, ce qui est exponentiellement plus grand que la taille de $\mathcal{S}$ puisque l'encodage de n nécessite seulement $\lfloor log_2(n+1) \rfloor$ bits.*

Bien que la transformation habituelle d'un VASS en un réseau de Petri ne préserve pas les propriétés structurelles, la vérification du caractère structurellement borné et de la terminaison structurelle en temps polynomial par une réduction en programme linéaire peut être étendue aux VASS comme nous le verrons dans le chapitre 11.

10.3 Propriétés semi-structurelles des réseaux de Petri

Lorsque l'on modélise un système par un réseau de Petri, on distingue souvent deux types de places :

— les *places de contrôles*, dont leurs marquages sont bornés et décrivent l'état actuel du système.

— les *places container*, dont les jetons représentent des ressources.

Il peut être intéressant de vérifier la terminaison (ou le caractère borné) du réseau de Petri en fixant le marquage initial des places de contrôle, mais avec un marquage initial arbitraire

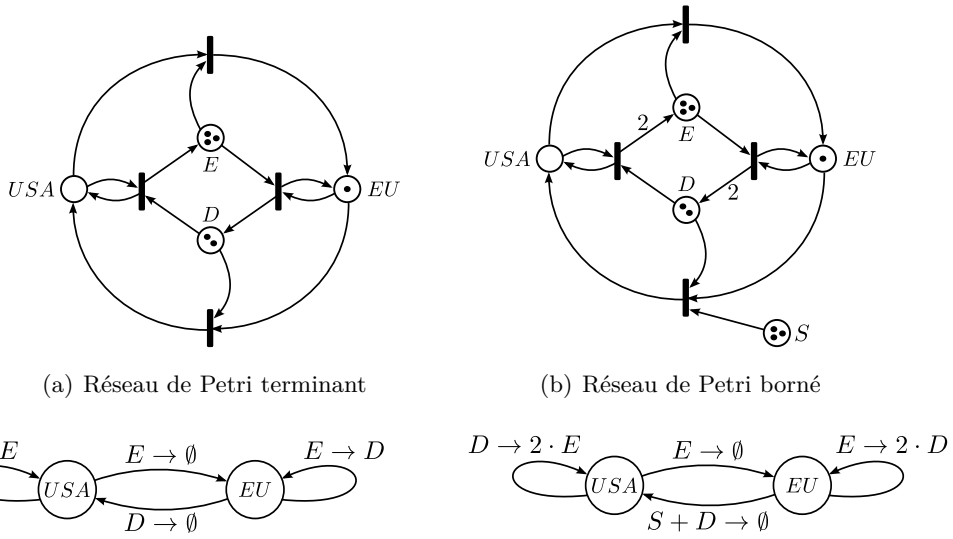
pour les places container. De cette façon, la terminaison semi-structurelle généralise à la fois la terminaison et la terminaison structurelle.

Une méthode immédiate nous permet de vérifier la terminaison semi-structurelle. Nous commençons par supprimer les places container et vérifions que le réseau de Petri résultant est borné. Nous déplions ensuite le réseau de Petri sous la forme d'un VASS et réincorporons les contraintes liées aux places container. Si le VASS obtenu est structurellement terminant, alors le réseau de Petri initial est semi structurellement terminant, c'est-à-dire qu'il termine quel que soit le marquage initial dans les places container. Rappelons que vérifier la terminaison d'un réseau de Petri requiert un espace exponentiel [100] alors que nous verrons dans le chapitre 11 que nous pouvons vérifier la terminaison structurelle d'un VASS en temps polynomial. Ainsi, considérer la terminaison semi-structurelle d'un réseau de Petri et donc la terminaison structurelle d'un VASS peut se révéler efficace pour vérifier si un réseau de Petri termine.

Exemple 10.3.1. *Considérons l'échangeur de monnaie modélisé par le réseau de Petri utilisant quatre places $P = \{E, D, EU, USA\}$ représenté sur la figure 10.3(a). Les places container E et D comptent respectivement le nombre d'euros et de dollars en notre possession. Un jeton se promène entre les deux états de contrôle EU et USA. Lorsque le jeton de contrôle se trouve dans EU alors les euros peuvent être changés en dollars, et lorsque le jeton de contrôle se trouve dans USA alors les dollars peuvent être changés en euros. Un déplacement du jeton de contrôle de EU vers USA (respectivement de USA vers EU) requiert le paiement d'une taxe en dollars (respectivement en euros). Ce réseau de Petri n'est pas structurellement terminant, car les monnaies peuvent circuler entre les euros et les dollars à condition qu'il y ait un jeton dans les places EU et USA. Considérons maintenant le dépliage du réseau de Petri en un VASS de deux états représentés sur la figure 10.3(c). Il est facile de voir que ce VASS est structurellement terminant. Ainsi, la modélisation de l'échangeur de monnaie de la figure 10.3(a) termine, quelle que soit la monnaie initiale que l'on possède.*

Une approche similaire peut être adoptée pour vérifier le caractère semi-structurellement borné d'un réseau de Petri. Considérons le réseau de Petri de la figure 10.3(b). Contrairement à l'exemple précédent, nous pouvons produire deux euros avec un dollar, et deux dollars avec un euro. Cependant, un déplacement de EU en USA consomme un jeton d'une nouvelle place S. Ce réseau de Petri n'est pas structurellement borné, car la quantité de monnaie peut ne pas être borné lorsque les places EU et USA possèdent chacun un jeton. Considérons les places E, D et S comme des places container. Le résultat du dépliage du réseau de Petri en VASS est représenté sur la figure 10.3(d). Ce VASS est structurellement borné. Ainsi, le réseau de Petri de la figure 10.3(b) est borné quel que soit le marquage initial des places E, D et S.

Remarquons que la modélisation du VASS de la figure 10.3(c) en un réseau de Petri selon la transformation classique correspond au réseau de Petri de la figure 10.3(a). Cet exemple donne une illustration des différences entre les réseaux de Petri et les VASS en ce qui concerne les propriétés structurelles. Un VASS peut être structurellement terminant, mais pas sa modélisation en tant que réseau de Petri.



(a) Réseau de Petri terminant

(b) Réseau de Petri borné

(c) Réseau de Petri terminant

(d) Réseau de Petri borné

FIGURE 10.3 – Modélisation d'un échangeur de monnaie.

10.4 Propriétés semi-structurelles des PNS

Les propriétés semi-structurelles sont également très intéressantes pour les PNS. Rappelons que les PNS peuvent représenter des MSG étendus (section 7.3). Ils peuvent contenir des variables entières positives permettant de modéliser un certain nombre de choses telles que des compteurs, des timers, des horloges, etc. Vérifier des propriétés (semi-)structurelles sur de tels MSG, telles que le caractère borné, devient alors beaucoup plus intéressant que la vérification de propriétés non-structurelles. Premièrement, au niveau de la complexité, les propriétés structurelles se résolvent beaucoup plus rapidement que les propriétés non-structurelles. Ensuite, le fait de savoir qu'un MSG est borné pour des valeurs de compteur spécifique ne nous garantit pas qu'en modifiant la valeur de ces compteurs le MSG reste borné. Prenons l'exemple du protocole simplifié des fenêtres coulissantes de la figure 10.4. Si nous fixons le nombre de jetons $d = 4$ et $x = 0$, le fait de savoir que le système est borné

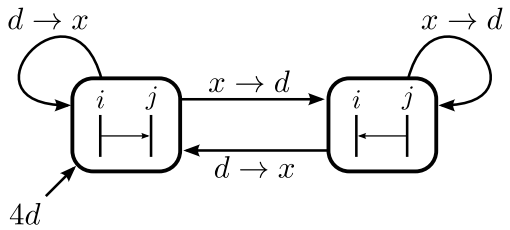


FIGURE 10.4 – Modélisation du protocole simplifié des fenêtres coulissantes avec un MSG étendu.

ne nous garantit pas que le protocole simplifié des fenêtres coulissantes ne diverge pas pour une taille de fenêtre plus grande. Si nous ne fixons que les places modélisant les instances, le fait que ce système est structurellement borné nous garantit que, quelle que soit la taille

de la fenêtre d, le protocole simplifié des fenêtres coulissantes ne diverge pas.

Notons que si nous fixons le marquage initial pour la totalité des places, alors la vérification d'une propriété est équivalant à sa vérification non-structurelle. La vérification d'une propriété semi-structurelle généralise ainsi la vérification de cette même propriété de manière non-structurelle toute en réduisant potentiellement la complexité de l'analyse suivant le nombre de places à traiter de manière abstraite. Une partie non négligeable de cette approche consiste à vérifier que le système dans lequel les places structurelles sont omises reste borné et à déplier ce système.

Chapitre

11 Vérification de propriétés structurelles de VASS

Dans ce chapitre, nous adaptons l'algorithme de Kosaraju et Sullivan [68] pour détecter des cycles pathologiques pour le caractère structurellement borné et la terminaison structurelle en temps polynomial. Le résultat de l'algorithme peut retourner un contre-exemple pour la propriété considérée sous la forme d'un multi-ensemble d'arcs qui correspond à un cycle pathologique.

11.1 Multi-ensemble d'arcs au lieu des cycles

Nous représenterons les cycles d'un VASS $\mathcal{S}$ comme des multi-ensembles particuliers d'arcs. Soit $x \in \mathbb{N}^A$ un multi-ensemble d'arcs. On note par $\|x\| = |\{a \in A \mid x[a] \geq 1\}|$ le nombre d'arcs distincts dans x et par A_x le *support* de x, c'est-à-dire l'ensemble des arcs $a \in A$ tel que $x[a] \geq 1$. Ainsi $\|x\| = |A_x|$. Le *graphe sous-jacent* G_x de x est le graphe (non dirigé) $G_x = (Q_x, E_x)$ où l'ensemble des sommets $Q_x = \{dom(a) \mid a \in A_x\} \cup \{cod(a) \mid a \in A_x\}$ collecte la source et la destination de tous les arcs dans x et l'ensemble des arêtes $E_x = \{\{dom(a), cod(a)\} \mid a \in A_x \text{ et } dom(a) \neq cod(a)\}$ conserve la trace de toutes les connexions induites par les arcs dans x.

Un multi-ensemble d'arcs $x \in \mathbb{N}^A$ est dit *connexe* si G_x est un graphe connexe. Soit $x \in \mathbb{N}^A$ et $C_1, ..., C_n \subseteq Q_x$ les composantes connexes de G_x. Pour chaque $1 \leq i \leq n$ et chaque $a \in A$, on pose $x_i[a] = x[a]$ si $dom(a) \in C_i$ et $x_i[a] = 0$ sinon. Alors $x = x_1 + ... + x_n$ et le multi-ensemble x_i est appelé une *composante connexe* de x.

Définition 11.1.1. *Un multi-ensemble d'arcs $x \in \mathbb{N}^A$ est dit* Eulérien *si pour chaque état $q \in Q$, on a $\sum_{dom(a)=q} x[a] = \sum_{cod(a)=q} x[a]$, c'est-à-dire qu'il y a autant d'arcs incidents que d'arcs sortants à chaque état. Un multi-ensemble d'arcs Eulérien et connexe est appelé une* circulation.

Notons que si x et y sont Eulériens, alors $x + y$ est Eulérien. Si de plus $x \leq y$, alors $y - x$ est également Eulérien.

Définition 11.1.2. *La* multiplicité *d'un multi-ensemble x dans un multi-ensemble y correspond au plus grand entier naturel k tel que $k \cdot x \leq y$.*

Chaque cycle $\gamma = a_1...a_n$ de $\mathcal{S}$ est représenté par un multi-ensemble d'arcs $x_\gamma = \sum_{i=1}^{i=n} a_i$, c'est-à-dire que $x_\gamma[a]$ est le nombre d'occurrences de a dans γ. Puisque γ est un cycle, le multi-ensemble d'arcs x_γ est non vide, Eulérien et connexe. Inversement, chaque circulation non vide correspond à un cycle de $\mathcal{S}$. Il s'agit d'une variante immédiate du théorème d'Euler [38, Th. 1.8.1] déjà évoqué dans les préliminaires (théorème 2.1.3).

Proposition 11.1.3. *(Théorème d'Euler) Soit $x \in \mathbb{N}^A$ une circulation non vide. Alors il existe un cycle γ tel que $x_\gamma = x$.*

Démonstration. Soit $\gamma = a_0...a_{l-1}$ un chemin dans $\mathcal{S}$ utilisant chaque arc $a \in A$ pas plus de $x[a]$ fois et de longueur maximale. Supposons que $dom(a_0) \neq cod(a_{l-1})$, c'est-à-dire que γ n'est pas un cycle. On pose $q = cod(a_{l-1})$ et on considère le multi-ensemble $x_\gamma \in \mathbb{N}^A$ de tous les arcs étiquetés présents dans γ. On a $x_\gamma \leq x$ et donc $\sum_{dom(a)=q} x_\gamma[a] \leq \sum_{dom(a)=q} x[a]$ et $\sum_{cod(a)=q} x_\gamma[a] \leq \sum_{cod(a)=q} x[a]$. Puisque $dom(a_0) \neq q$, on a $\sum_{cod(a)=q} x_\gamma[a] = 1 + \sum_{dom(a)=q} x_\gamma[a]$. Comme x est Eulérien, on a $\sum_{dom(a)=q} x[a] = \sum_{cod(a)=q} x[a]$. Il s'ensuit que $\sum_{dom(a)=q} x_\gamma[a] < \sum_{dom(a)=q} x[a]$. Ainsi $x_\gamma[a] < x[a]$ pour un arc $a \in A$ tel que $dom(a) = q$. Nous pouvons alors ajouter une occurrence de a à la fin de γ impliquant que la longueur de γ n'est pas maximale. Ainsi, $dom(a_0) = cod(a_{l-1})$ et γ est un cycle.

Nous procédons encore par contradiction. Supposons que le cycle $a_0...a_{l-1}$ ne satisfait pas $x_\gamma = x$. Alors il existe des $a \in A$ tels que $x[a] > x_\gamma[a]$. Comme G_x est connexe, on peut supposer que $dom(a) = dom(a_i)$ pour certain i, ou $cod(a) = dom(a_i)$ pour certain i. Dans les deux cas, on peut construire un chemin plus long que γ en ajoutant une occurrence de a, ce qui contredit à nouveau la maximalité de γ. □

Il est facile de montrer que le problème du caractère structurellement borné et le problème de la terminaison structurelle sont des problèmes NP : un certificat d'un cycle pathologique est simplement un sous-ensemble d'arc $C \subseteq A$ pour lequel il existe un multi-ensemble Eulérien d'arcs x tels que $A_x = C$ et $cout(x) \gneq \vec{0}$ (respectivement $cout(x) \geq \vec{0}$). Ceci peut être vérifié en temps polynomial en utilisant la programmation linéaire car s'il existe une solution rationnelle pour ce système alors il existe également une solution entière.

Remarque 11.1.4. *Considérons un chemin pathologique γ représenté par une circulation $x \in \mathbb{N}^A$. Soit d le plus grand diviseur commun de tous les coefficients de x. Alors le multi-ensemble $x' = 1/d \cdot x \in \mathbb{N}^A$ est une circulation non vide qui est plus petite que x mais qui représente aussi un cycle pathologique. Comme d peut être calculé en temps polynomial (avec l'algorithme d'Euclide), la circulation x' peut être obtenue facilement à partir de x afin de simplifier la représentation du bug.*

11.2 Calculer un cycle pathologique comme un multi-ensemble d'arcs

Dans [68], Kosaraju et Sullivan montrent comment détecter un cycle avec un coût de zéro en temps polynomial à l'aide d'une réduction vers une série de programmes linéaires. Nous expliquons ici pourquoi cette technique peut être adaptée sans effort pour vérifier la terminaison structurelle ou le caractère structurellement borné. En effet, il suffit de remplacer l'égalité du vecteur $x = \vec{0}$ par $x \geq \vec{0}$ ou $x \gneq \vec{0}$ respectivement dans les programmes linéaires considérés par leur algorithme. Nous donnons ci-dessous l'algorithme résultant pour la terminaison structurelle seulement.

Afin de détecter un cycle pathologique pour la terminaison structurelle dans un VASS $\mathcal{S} = (Q, A)$, la première étape consiste à calculer le sous-ensemble $A' \subseteq A$ qui rassemble tous les arcs apparaissant dans un multi-ensemble $\mathscr{F}$ de cycles tel que $cout(\mathscr{F}) \geq \vec{0}$. Pour

ce faire, pour chaque arc $b \in A$, on considère le programme linéaire ci-dessous avec $|A|$ variables rationnelles $x \in \mathbb{Q}^A$.

$$(P_b) \begin{cases} \sum_{cod(a)=q} x[a] = \sum_{dom(a)=q} x[a] \text{ pour chaque } q \in Q \\ x[a] \geq 0 \text{ pour chaque } a \in A \\ x[b] \geq 1 \\ \sum_{a \in A} x[a] \cdot cout(a) \geq \vec{0} \end{cases}$$

Il est clair que b appartient à A' si, et seulement si, P_b a une solution. Dans ce cas nous désignions par x_b une solution à ce problème avec $x_b \in \mathbb{N}^A$. Ensuite, trois cas se présentent :

1. Si A' est vide alors $\mathcal{S}$ est structurellement terminant, car aucun arc n'appartient au cycle pathologique.
2. Si A' est non vide et fortement connexe, alors $\mathcal{S}$ est non-structurellement terminant, car le multi-ensemble d'arcs $x = \sum_{b \in A'} x_b$ est connexe, Eulérien et $cout(x) \geq \vec{0}$. Notons ici que le support A_x de x est maximal, car $A_x = A'$
3. Soit $A_1, ..., A_n$ les composantes fortement connexes de A' avec $n \geq 2$. On considère les n VASS $\mathcal{S}_1, ..., \mathcal{S}_n$ obtenus à partir de $\mathcal{S}$ par une réduction aux sous-ensembles d'arcs $A_1, ..., A_n$ respectivement. Il est alors clair que $\mathcal{S}$ admet un cycle pathologique si, et seulement si, un des n systèmes $\mathcal{S}_1, ..., \mathcal{S}_n$ admet un cycle pathologique. Ainsi, il suffit d'appliquer récursivement l'algorithme à chacun de ces systèmes.

La complexité en temps pour l'algorithme est de $O(Z|A||Q|)$ avec Z la complexité en temps pour le programme linéaire P_b. Notons ici que nous pouvons exiger que cet algorithme retourne une circulation qui corresponde à un cycle pathologique si le système n'est pas structurellement terminant. Nous devons simplement obtenir une solution entière $x' \in \mathbb{N}^A$ au problème P_b à partir de la solution rationnelle $x \in \mathbb{Q}^A$. Pour ce faire, on peut utiliser l'algorithme d'Euclide pour calculer le plus petit multiple commun m des dénominateurs de tous les coefficients de x puis considérer $x' = m \cdot x$.

11.3 Étude expérimentale

Reprenons l'exemple du protocole simplifié des fenêtres coulissantes de la section 9.5. Soit $\mathcal{S}$ le PNS modélisant ce protocole. Nous commençons par considérer le PNS $\mathcal{S}^\circ$ modélisant les marquages préfixes accessibles en utilisant la transformation décrite dans la section 9.1. Nous déplions ensuite le PNS $\mathcal{S}^\circ$ en un PNS $\mathcal{S}'$ afin de faire disparaître les places correspondant aux instances comme décrit dans la section 10.4. Pour terminer, nous vérifions le caractère structurellement borné du PNS $\mathcal{S}'$ en utilisant l'algorithme de Kosaraju et Sullivan [68].

Si le PNS $\mathcal{S}'$ est structurellement borné, alors le PNS $\mathcal{S}^\circ$ est semi-structurellement borné. Si le PNS $\mathcal{S}^\circ$ est semi-structurellement borné, alors le PNS $\mathcal{S}$ est préfixe borné pour toutes les tailles de fenêtre. Ainsi, si le PNS $\mathcal{S}'$ est structurellement borné, alors le protocole simplifié des fenêtres coulissantes ne diverge pas, quelle que soit la taille des fenêtres.

Nous comparons les résultats obtenus par cette méthode avec les résultats obtenus précédemment pour une taille de fenêtres fixée.

n (taille de fenêtre)	Préfixe-borné (avec TINA)	Non-divergence (avec Minisat2)	Semi-structurellement préfixe borné (Kosaraju et Sullivan avec GLPK [1])
40	9	<0.01	0.6
60	37	<0.01	0.6
80	110	<0.01	0.6
100	256	<0.01	0.6
80 000		5.9	0.6
160 000		14	0.6

FIGURE 11.1 – Temps d'exécution en sec. pour le protocole simplifié des fenêtres coulissantes.

Comme nous pouvions nous y attendre, le temps nécessaire pour vérifier le caractère semi-structurellement préfixe borné du protocole ne dépend de la taille des fenêtres. De plus, la vérification du caractère semi-structurellement préfixe borné est beaucoup plus rapide que la vérification du caractère borné. Elle parvient même à rivaliser avec la vérification de la divergence de MSG.

Ces résultats confirment l'intérêt des propriétés semi-structurelles. D'une part l'information apportée par ce type de propriété est très intéressante, voire même plus intéressante que les propriétés non structurelles. Pour le protocole des fenêtres coulissantes, en particulier, il est plus intéressant de certifier que le protocole ne diverge pas, quelle que soit la taille des fenêtres, plutôt que de certifier que le protocole ne diverge pas pour une certaine taille de fenêtre. Le second intérêt des propriétés semi-structurelles est la réduction du temps de calcul et de la complexité comparée aux propriétés non structurelles.

Chapitre

12 Construire un multi-ensemble de lassos élémentaires

Dans ce chapitre, nous nous proposons de décrire les cycles pathologiques d'un VASS sous la forme d'un multi-ensemble de cycles particuliers que nous appelons des *lassos élémentaires*. De manière approximative, un lasso élémentaire de valuation k est un cycle constitué de k itérations d'un cycle élémentaire ainsi que d'un chemin élémentaire aller et un chemin élémentaire retour d'un état initial vers un nœud du cycle.

L'algorithme pour détecter des chemins pathologiques, décrit à la section 11.2, peut nous fournir des circulations qui correspondent à des cycles pathologiques. Afin d'aider la compréhension d'un bug structurel détecté par une circulation, il est utile de représenter ce contre-exemple sous la forme d'un cycle pathologique. Cependant, la longueur minimale d'un cycle pathologique peut être exponentielle dans la taille du VASS comme illustré dans l'exemple suivant.

Exemple 12.0.1. *Considérons le VASS avec un unique état et six arcs étiquetés par les six vecteurs 6-dimensionnels suivant :*

$$\begin{array}{rcrrrrrr} t_1 = & (& 2, & 0, & 0, & 0, & 0, & -1 \ ) \\ t_2 = & (& -1, & 2, & 0, & 0, & 0, & 0 \ ) \\ t_3 = & (& 0, & -1, & 2, & 0, & 0, & 0 \ ) \\ t_4 = & (& 0, & 0, & -2, & 1, & 0, & 0 \ ) \\ t_5 = & (& 0, & 0, & 0, & -2, & 1, & 0 \ ) \\ t_6 = & (& 0, & 0, & 0, & 0, & -2, & 1 \ ) \end{array}$$

Il est facile de voir que tout cycle pathologique de ce VASS $\mathcal{S}$ a besoin de tous les arcs en raison de leurs dépendances. De plus, un cycle pathologique qui contient une occurrence de t_6 nécessite la présence de 2 occurrences de t_5, 4 occurrences de t_4 et donc 4 occurrences de t_3, 2 occurrences de t_2 et une occurrence de t_1. Ainsi, le cycle pathologique $\gamma = t_1 + 2 \cdot t_2 + 4 \cdot t_3 + 4 \cdot t_4 + 2 \cdot t_5 + t_6$ est de longueur minimale. Nous pouvons facilement généraliser cet exemple constitué de $2 \times m$ arcs avec des cycles pathologiques de longueur minimale $2 \times (2^m - 1)$.

Ainsi, l'énumération des arcs apparaissant le long d'un cycle pathologique est irréaliste en général. Par conséquent, nous avons besoin d'une *représentation compacte* des cycles pathologiques.

Pour le cas particulier des VASS à un seul état, une circulation peut être vue comme un multi-ensemble de cycles avec un état initial, et ainsi comme une représentation simple et compacte d'un cycle pathologique. Dans ce travail, nous voulons étendre cette propriété à n'importe quel VASS. Nous introduirons par la suite une classe particulière de cycles appelée lassos. Les lassos sont constitués de trois chemins consécutifs de longueur $\leq |Q|$.

Nous montrons comment calculer un cycles pathologiques sous la forme d'un multi-ensemble de lassos en temps polynomiale. De plus, le nombre de lassos dans le multi-ensemble sera $\leq p$.

Définition 12.0.2. *Soit $q, q' \in Q$ deux états de $\mathcal{S}$. Soit γ_0 un cycle de $\mathcal{S}$ débutant de q'. Soit γ_1 un chemin de q vers q' et γ_2 un chemin de q' vers q. Soit $k \in \mathbb{N}$. Nous supposons que la longueur de chaque chemin γ_0, γ_1 et γ_2 est au plus égale au nombre d'états $|Q|$. Soit $W = \gamma_1 \cdot \gamma_0^k \cdot \gamma_2$ le cycle démarrant de q et constitué de γ_1, suivi par k itérations du cycle γ_0, suivi par γ_2. Si la longueur de W est d'au moins 1, alors W est appelé un* lasso *de* valuation *k. Un lasso est dit* élémentaire *si les trois chemins γ_0, γ_1 et γ_2 dont il est constitué sont élémentaires.*

Un lasso est souvent représenté par un multi-ensemble d'arcs $W = D + k \cdot C$ où D est le multi-ensemble d'arcs présents dans γ_1 et γ_2, et C est l'ensemble des arcs apparaissant dans γ_0. Alors le multi-ensemble est connexe et Eulérien. Notons que si le lasso est élémentaire alors le cycle $\gamma_1 \cdot \gamma_2$ contient au maximum deux fois un même arc.

Exemple 12.0.3. *Continuons l'exemple 10.2.2 avec $p = 2$. Nous avons observé que le coût du cycle γ est $cout(\gamma) = (1, 4)$. Considérons les deux lassos élémentaires $W_1 = a_1 \cdot l_1^{10} \cdot a_2 \cdot a_3$ de valuation 10 et $W_2 = a_1 \cdot a_2 \cdot l_2^6 \cdot a_3$ de valuation 6. Il est facile de voir que $2 \cdot cout(\gamma) = cout(W_1) + cout(W_2)$.*

Cet exemple illustre précisément comment des lassos élémentaires peuvent représenter un cycle à un facteur multiplicatif près. Le but de ce chapitre est de montrer que l'on peut construire une telle représentation en général à partir d'un cycle donné sous la forme d'un multi-ensemble d'arcs $\hat{H}$ connexe et Eulérien. Les algorithmes développés dans ce chapitre nous permettent de calculer en temps polynomial à partir du multi-ensemble $\hat{H}$ un multi-ensemble $\mathscr{F}$ constitué d'au plus p lassos élémentaires distincts démarrant d'un même état tel que $cout(\mathscr{F}) = m \cdot cout(\hat{H})$ pour un certain $m \in \mathbb{N}^*$ (théorème 12.3.2).

Dans le reste de ce chapitre, nous fixons une circulation non vide $\hat{H} \in \mathbb{N}^A$ avec $cout(\hat{H}) \geq \vec{0}$ sur les états $Q_{\hat{H}}$ et fixons un état $\hat{q} \in Q_{\hat{H}}$. De plus, si C est un multi-ensemble d'arcs, nous noterons Q_C son ensemble d'états et A_C son ensemble d'arcs.

12.1 Trouver un cycle élémentaire adapté dans une circulation

L'ingrédient principal de cette construction de lassos élémentaires à partir d'une circulation $\hat{H}$ consiste à rechercher dans $\hat{H}$ un cycle élémentaire $C \leq \hat{H}$ avec une *multiplicité* $k \geq 1$ dans $\hat{H}$ tel que le multi-ensemble $\hat{H} - k \cdot C$ est connexe et contient l'état fixé $\hat{q}$ s'il n'est pas vide. Pour ce faire, nous utilisons la définition suivante.

Définition 12.1.1. *Soit $H \in \mathbb{N}^A$ un multi-ensemble d'arcs, $\sigma \in A^*$ un chemin, et $q_0 \in Q$ un état de $\mathcal{S}$. Un cycle C de multiplicité $k \geq 1$ dans H est* adéquat *pour H, σ et q_0 s'il satisfait les deux conditions suivantes :*

1. *Chaque composante connexe de $H - k \cdot C$ contient un état de $Q_\sigma \cup \{q_0\}$.*
2. *Il existe un arc $a \in A_C \setminus A_\sigma$ tel que $H(a) = k$.*

Par la suite, nous appliquerons l'algorithme 1 afin de construire un cycle élémentaire C qui est adéquat pour $\mathscr{F}$, le chemin vide $\emptyset$, et l'état fixé $\hat{q}$ (propriété 12.1.2). En conséquence, nous obtenons que le multi-ensemble $\mathscr{F} - k \cdot C$ est connexe et contient l'état fixé $\hat{q}$ s'il n'est pas vide.

Algorithme 1 CycleÉlémentaireAdéquat(H, σ, q_0)

Données : $H \in \mathbb{N}^A$ est une circulation non vide.
Données : σ un chemin élémentaire contenant des arcs de A et tel que σ n'est pas un cycle.
Données : $q_0 \in Q_H$ et $q_0 \in Q_\sigma$ si le chemin σ n'est pas vide.
si il existe un cycle élémentaire $C \leq H$ avec $Q_C \cap (Q_\sigma \cup \{q_0\}) = \emptyset$ **alors**
  Soit C un tel cycle élémentaire et k la multiplicité de C dans H
  si chaque composante connexe de $H - k \cdot C$ contient un état de $Q_\sigma \cup \{q_0\}$ **alors**
    renvoyer C # En particulier si $H = k \cdot C$.
  sinon
    Soit H' une composante connexe de $H - k \cdot C$ avec $Q_{H'} \cap (Q_\sigma \cup \{q_0\}) = \emptyset$.
    Soit q_0' un état de $Q_{H'} \cap Q_C$ et a un arc de A_C avec $H(a) = k$.
    Soit σ' un chemin fait de tous les arcs de $A_C \setminus \{a\}$
    renvoyer CycleÉlémentaireAdéquat(H', σ', q_0')
  fin si
sinon
  Soit $b \in A_H \setminus A_\sigma$
  $\beta \leftarrow b$ # Initialement, β est un chemin de longueur 1
  tant que β ne contient pas de cycle élémentaire **faire**
    si il existe un arc existe $b' \in A_H \setminus A_\sigma$ avec $dom(b') = cod(b)$ **alors**
      Choisir un arc $b' \in A_H \setminus A_\sigma$ avec $dom(b') = cod(b)$
    sinon
      Choisir un arc $b' \in A_H \cap A_\sigma$ tel que $dom(b') = cod(b)$
    fin si
    $b \leftarrow b'$
    Ajouter l'arc b à la fin du chemin β
  fin tant que
  renvoyer le cycle élémentaire C contenu dans β
fin si

L'algorithme 1 procède de la manière suivante. Supposons que $H \in \mathbb{N}^A$ soit une circulation non vide et $\sigma = a_1 ... a_n$ un chemin élémentaire constitué d'arc de A tel que σ n'est pas un cycle. Soit $q_0 \in Q_H$ un état de H tel que $q_0 \in Q_\sigma$ si σ n'est pas vide. Nous considérons le sous-ensemble $A' \subseteq A$ constitué de tous les arcs de A_H dont les sources et les destinations n'appartiennent pas à $Q_\sigma \cup \{q_0\}$. Soit $A'_1, ..., A'_n$ les composantes fortement connexes de A'. Alors il existe un cycle élémentaire C dans H avec $Q_A \cap (Q_\sigma \cup \{q_0\})) = \emptyset$ si, et seulement si, une des composantes A'_i a deux états ou A' un arc bouclant. Selon que cette condition soit vérifiée, nous considérons l'un ou l'autre de ces cas :

1. Si la condition est satisfaite, nous fixons un tel cycle élémentaire C et considérons sa multiplicité $k \geq 1$ dans H. Si chaque composante connexe de $H - k \cdot C$ contient au

moins un état de $Q_\sigma \cup \{q_0\}$ alors C est adéquat pour H, σ et q_0. Supposons maintenant que $H - k \cdot C$ est non vide et considérons des composantes connexes H' de $H - k \cdot C$ qui ne contiennent pas d'état de $Q_\sigma \cup \{q_0\}$. Soit $a \in A_\gamma$ tel que $H(a) = k$. Alors $H'(a) = 0$ et donc $\|H'\| < \|H\|$. Par ailleurs, on peut choisir un état $q'_0 \in Q_{H'} \cap Q_\gamma$ et construire un chemin σ' à partir de tous les arcs de $A_\gamma \setminus \{a\}$. Alors σ' contient tous les arcs de $A_\gamma \cap A_{H'}$ et $q'_0 \in Q_{\sigma'}$ des que σ' n'est pas vide. Par ailleurs, σ' n'est pas un cycle. Nous affirmons que tout cycle élémentaire adéquat pour H', σ' et q'_0 est également adéquat pour H, σ et q_0. Notons ici que la terminaison de l'algorithme 1 est garantie par $\|H'\| < \|H\|$

2. Supposons maintenant qu'il n'existe pas de tel cycle élémentaire. Comme σ n'est pas un cycle et H est une circulation non vide, nous pouvons choisir un arc $b \in A_H \setminus A_\sigma$ arbitrairement et considérer le chemin $\beta = b$. Alors le chemin est construit itérative ment par l'ajout d'arcs de A_H à la fin de β jusqu'à ce que β contienne un cycle élémentaire. À chaque itération, il y a un candidat potentiel pour compléter β, car H est une circulation. Cependant, nous avons besoin que les arcs de $A_H \setminus A_\sigma$ soient prioritaires. De toute évidence, cette boucle termine après un maximum de $|Q_H|$ itérations. La propriété 12.1.2 affirme que tout cycle élémentaire dans β est adéquat pour H, σ et q_0.

Proposition 12.1.2. *Soit $H \in \mathbb{N}^A$ une circulation. Soit $q_0 \in Q_H$ un état de H et σ un chemin élémentaire contenant des arcs de A tel que $q_0 \in Q_\sigma$ si la longueur de σ est ≥ 1. Nous supposons également que σ n'est pas un cycle élémentaire. L'algorithme 1 retourne un cycle élémentaire $C \leq H$ qui est adéquat pour H, σ et q_0.*

Démonstration. Notons ici que $q_0 \in Q_\sigma$ lorsque la longueur de σ est positive. D'autre part, Q_σ est vide si σ est vide. Ce cas spécial est effectivement appliqué dans l'algorithme 2. Procédons par induction sur le nombre d'arcs dans H. Supposons $\|H\| = 1$. Comme H contient un unique arc, celui-ci forme une boucle et donc un cycle élémentaire C. Soit k la multiplicité de C dans H. Le multi-ensemble $H - k \cdot C$ est vide : il n'y a pas de composante connexe. Ainsi, C satisfait les deux exigences. Étape d'induction : nous distinguons deux cas.
Cas 1 : Il y a un cycle élémentaire $C \leq H$ ne contenant aucun nœud de $Q_\sigma \cup \{q_0\}$. Soit $k \geq 1$ la multiplicité de C dans H. Nous distinguons deux sous cas :

1. Chaque composante connexe de $H - k \cdot C$ contient un état de $Q_\sigma \cup \{q_0\}$. Cela vaut en particulier si $H - k \cdot C$ est vide. Comme k est la multiplicité de C dans H, il y a un arc $a \in A_C$ tel que $H(a) = k$. Comme C ne contient pas d'état de $Q_\sigma \cup \{q_0\}$, l'arc a n'appartient pas à A_σ. Par conséquent, le cycle élémentaire C remplit les deux exigences.

2. Le multi-ensemble $H - k \cdot C$ n'est pas vide et il existe une composante connexe H' de $H - k \cdot C$ qui ne contient pas d'état de $Q_\sigma \cup \{q_0\}$ (figure 12.1(a)). Nous avons $\|H'\| \leq \|H - k \cdot C\|$, car k est la multiplicité de C dans H. Alors $Q_{H'} \cap Q_C \neq \emptyset$ sinon il n'y aurait pas de chemin provenant de l'ensemble des états de $Q_{H'}$ vers l'ensemble des états Q_C dans H. Soit $q'_0 \in Q_{H'} \cap Q_C$. Soit $a \in A_C$ tel que $H(a) = k$. Nous considérons

le chemin élémentaire σ' constitué de tous les arcs de $A_C \setminus \{a\}$. Alors σ' contient tous les arcs de $A_{H'} \cap A_C$ et $q'_0 \in Q_{\sigma'}$ si σ' n'est pas vide. De plus, $q'_0 \in Q_{H'}$ et σ' ne sont pas des cycles élémentaires. Par hypothèse d'induction, $CycleÉlémentaireAdequat(H', \sigma', q'_0)$ retourne un cycle élémentaire C' de multiplicité $k' \geq 1$ dans H' tel que :

— Chaque composante connexe de $H' - k' \cdot \sigma'$ contient un état de $Q_{\sigma'} \cup \{q'_0\}$.

— Il existe un arc $a' \in A_{C'} \setminus A_{\sigma'}$ tel que $H'(a') = k'$.

Ceci est illustré sur la figure 12.1(b). Comme σ' contient tous les arcs de C présent dans H', nous avons $a' \notin A_C$. Par conséquent, $H(a') = (H - k \cdot C)(a')$ et donc $H'(a') = H(a')$. Ainsi, k' est également la multiplicité de C' dans H. Nous prouvons que $H - k' \cdot C'$ est connexe. Comme $H - k \cdot C \geq k' \cdot C'$, nous avons $H - k' \cdot C' \geq k \cdot C \geq C$. Soit $q'' \in Q_{H - k' \cdot C'}$. Nous observons qu'il existe un chemin de q'' vers un état de C constitué d'arcs de $H - k' \cdot C'$. Ceci est trivial si $q'' \in Q_C$. Si $q'' \notin Q_C$ alors q'' appartient à une des composantes connexes de $H - k \cdot C$. Nous distinguons deux cas :

— $q'' \in Q_{H'} \setminus Q_{C'}$. Alors $q'' \in Q_{H' - k' \cdot C'}$. Comme chaque composante connexe de $Q_{H' - k' \cdot C'}$ contient un état de $Q_{\sigma'} \cup \{q'_0\}$ et $Q_{\sigma'} \cup \{q'_0\} \subseteq Q_C$, alors il existe un chemin allant de q'' vers C dans $H' - k' \cdot C'$ et donc dans $H - k' \cdot C'$.

— $q'' \in Q_{H''}$ avec H'' une composante connexe de $H - k \cdot C$ différent de H'. Alors $Q_{H''} \cap Q_C \neq \emptyset$ sinon il ne serait pas un chemin allant d'un état de $Q_{H''}$ vers un état de Q_C dans H. Ainsi, il existe un chemin allant de q'' vers C dans H'' et donc dans $H - k' \cdot C'$.

Ainsi, $H - k' \cdot C'$ est connexe. Comme $q_0 \in H$ et $q_0 \notin H'$, q_0 est dans $H - k' \cdot C'$. Comme H' ne contient pas d'état de $Q_\sigma \cup \{q_0\}$, C' ne contient pas d'état de $Q_\sigma \cup \{q_0\}$ non plus. Rappelons maintenant que H contient l'état q_0 et donc $H - k' \cdot C'$ contient également q_0. Comme H' ne contient pas d'état de $Q_\sigma \cup \{q_0\}$, nous avons $a' \in A_{C'} \setminus A_\sigma$. Ainsi, le cycle élémentaire C' répond aux deux exigences (figure 12.1(c)).

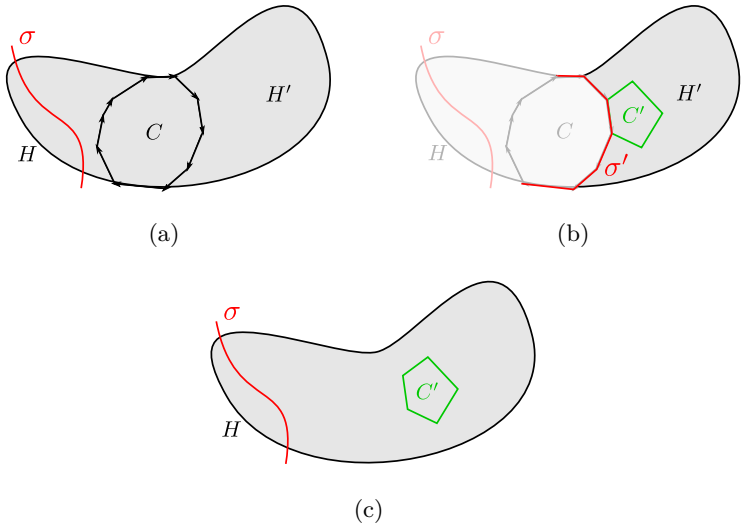


FIGURE 12.1 – Illustration pour la preuve de la propriété 12.1.2.

Cas 2 : Tout cycle élémentaire dans H contient au moins un état de $Q_\sigma \cup \{q_0\}$. Comme H est Eulérien et σ est un chemin élémentaire non fermé, il existe un arc $b \in A_H \setminus A_\sigma$. L'algorithme 1 construit un cycle élémentaire $C = a_0 a_1 ... a_{n-1}$ de multiplicité $k \geq 1$ dans H utilisant de préférence les arcs qui n'apparaissent pas dans σ. Comme le chemin élémentaire σ n'est pas fermé, le cycle élémentaire C ne peut pas être composé seulement des arcs de σ. Ainsi, C contient au moins un arc de $A_C \setminus A_\sigma$. Supposons que $a_i \in A_\sigma \cap A_C$. Comme nous utilisons une priorité sur les arcs, l'arc a_i est le seul arc avec $dom(a_i) = cod(a_{i-1}(mod\ n))$. Comme H est Eulérien, nous avons $H(a_{i-1(mod\ n)}) \leq H(a_i)$. Comme C contient au moins un arc de $A_C \setminus A_\sigma$, il existe $a \in A_C \setminus A_\sigma$ tel que $H(a) = k$.

Comme H est Eulérien, $H - k \cdot C$ est Eulérien. Soit H' une composante connexe de $H - k \cdot C$. Comme $H - k \cdot C$ est Eulérien, H' est Eulérien. Ainsi, il existe un cycle élémentaire dans H' et donc H' contient un état de $Q_\sigma \cup \{q_0\}$. Ainsi, toutes les composantes connexes de $H - k \cdot C$ contient un état de $Q_\sigma \cup \{q_0\}$. □

12.2 Construire un multi-ensemble de lassos élémentaires à partir d'un multi-ensemble d'arcs

La construction d'un multi-ensemble $\mathscr{F}$ de lassos élémentaires représentatif d'une circulation $\hat{H}$ donnée est décrite dans l'algorithme 2. Celui-ci effectue itérativement l'étape suivante. Premièrement, un cycle élémentaire adéquat $C = CycleÉlémentaireAdequat(\hat{H}, \emptyset, \hat{q})$ est trouvé en utilisant l'algorithme 1. Soit k la multiplicité de C dans $\hat{H}$. Alors le multi-ensemble Eulérien $\hat{H} - k \cdot C$ est connexe (propriété 12.1.2). De plus, $\|\hat{H} - k \cdot C\| < \|\hat{H}\|$ et $\hat{q} \in Q_{\hat{H}-k\cdot C}$ à condition que $\hat{H} - k \cdot C$ ne soit pas vide. Si $\hat{q}$ apparaît dans C alors $k \cdot C$ est un lasso élémentaire W débutant de $\hat{q}$. Supposons que $\hat{q} \notin Q_C$. Soit q un état de C. Comme $\hat{H}$ est connexe, il y a un chemin élémentaire σ_1 allant de $\hat{q}$ vers q et un chemin élémentaire σ_2 allant de q vers $\hat{q}$. Soit D le multi-ensemble d'arcs correspondant au cycle $\sigma_1\sigma_2$. Alors le multi-ensemble d'arcs $W = D + k \cdot C$ représente un lasso élémentaire démarrant de $\hat{q}$. De plus, $W \leq 3 \cdot \hat{H}$ puisque chaque arc apparaît au plus 2 fois dans $\sigma_1\sigma_2$. Nous distinguons alors trois cas :

1. Si $W = \hat{H}$ alors le lasso élémentaire W est ajouté à $\mathscr{F}$ et retiré de $\hat{H}$ menant ainsi au multi-ensemble vide.

2. Si $W \leq \hat{H}$, $\hat{H} - W$ est connexe et $\hat{q} \in Q_{\hat{H}-W}$ alors le lasso élémentaire W est ajouté à $\mathscr{F}$ et retiré de $\hat{H}$ menant ainsi au nouveau multi-ensemble $H' = \hat{H} - W$ avec $\hat{q} \in Q_{H'}$. Comme k est la multiplicité de C dans $\hat{H}$, nous avons $\|H'\| < \|\hat{H}\|$.

3. Dans le dernier cas, le multi-ensemble $\mathscr{F}$ est multiplié par 3 et le multi-ensemble d'arcs $\hat{H}$ est également multiplié par 3, menant à un nouveau multi-ensemble d'arcs H. Nous considérons le nouveau lasso $W' = D + k' \cdot C$ avec k' la multiplicité de C dans $H - D$. Alors $k \leq k'$ et $W' \leq H$. Le lasso W' est ajouté à $\mathscr{F}$ et retiré de H menant à un nouveau multi-ensemble H'. À ce stade, nous affirmons que $A_{H'} = A_{\hat{H}-k\cdot C}$. Ainsi, H' est connexe, $\hat{q} \in Q_{H'}$ et $\|H'\| < \|\hat{H}\|$

Algorithme 2 Construction du multi-ensemble de lassos élémentaire

Données : Une circulation non vide $\hat{H}$ et un état $\hat{q} \in Q_{\hat{H}}$
$\mathscr{F} \leftarrow \emptyset$ # Initialement, $\mathscr{F}$ est un multi-ensemble de lassos élémentaires vide
$H \leftarrow \hat{H}$ # Initialement, $cout(\mathscr{F}) + cout(H) = cout(\hat{H})$
tant que $H \neq \emptyset$ **faire**
$C \leftarrow$ CycleÉlémentaireAdéquat$(H, \emptyset, \hat{q})$ # On a $C \leq H$
Soit k la multiplicité de C dans H # Nous obtenons que $H - k \cdot C$ est connexe
si $\hat{q} \in Q_C$ **alors**
$D \leftarrow \emptyset$ # $D \in \mathbb{N}^A$ est un multi-ensemble d'arcs vide
$W \leftarrow k \cdot C$ # Le multi-ensemble W représente un lasso élémentaire tel que $W \leq \hat{H}$
sinon
Soit q un état de Q_C.
Soit γ_1 un chemin élémentaire allant de $\hat{q}$ à q formé par des arcs de A_H.
Soit γ_2 un chemin élémentaire allant de q à $\hat{q}$ formé par des arcs de A_H.
Soit D un multi-ensemble d'arcs correspondant au cycle $\gamma_1\gamma_2$. # Alors $D \leq 2 \cdot H$
$W \leftarrow D + k \cdot C$ # le multi-ensemble W représente un lasso élémentaire tel que $W \leq 3 \cdot H$
fin si
si $(H = W)$ ou $(H \gneq W$ et $H - W$ est connexe et $\hat{q} \in Q_{H-W})$ **alors**
Ajouter le lasso élémentaire $W = D + k \cdot C$ à $\mathscr{F}$.
$H \leftarrow H - W$
sinon
Soit k' la multiplicité de C dans $3 \cdot H - D$ # Ici, $D + k' \cdot C$ représente un lasso élémentaire
$W' \leftarrow D + k' \cdot C$ # Nous avons $k' \geq k$ et $A_{H-k\cdot C} = A_{3\cdot H - W'}$
$\mathscr{F} \leftarrow 3 \cdot \mathscr{F}$
Ajouter le lasso élémentaire $W' = D + k' \cdot C$ à $\mathscr{F}$.
$H \leftarrow H - W'$
fin si
fin tant que
renvoyer $\mathscr{F}$

Ainsi, dans tous les cas, nous obtenons que H' est Eulérien et connexe. De plus, $\hat{q} \in Q_{H'}$ à condition que H' ne soit pas vide et donc la prochaine itération de l'algorithme peut être menée dans les mêmes conditions. Par ailleurs, comme $\|H'\| < \|\hat{H}\|$ l'algorithme 2 termine après au maximum $|A|$ itérations.

Il est clair que la propriété $cout(\mathscr{F}) + cout(H) = m \cdot cout(\hat{H})$ pour un $m \in \mathbb{N}^*$ est l'invariant de boucle de l'algorithme 2.

Proposition 12.2.1. *Soit $\hat{H}$ une circulation non vide et $\hat{q} \in Q_{\hat{H}}$ un état de $\hat{H}$. L'algorithme 2 retourne un multi-ensemble non vide $\mathscr{F}$ de lassos élémentaires partant de $\hat{q}$ tel que $cout(\mathscr{F}) = m \cdot cout(\hat{H})$ pour un certain $m \in \mathbb{N}^*$.*

Démonstration. La propriété $cout(\mathscr{F}) + cout(H) = m \cdot cout(\hat{H})$ pour $m \in \mathbb{N}^*$ est l'invariant de boucle de l'algorithme 2. □

Soulignons que $\mathscr{F}$ est constitué d'au plus $|A|$ lassos car un lasso est ajouté à chaque itérations. Par ailleurs, la valuation de chaque lasso de $\mathscr{F}$ est inférieure à $3^{|A|} \times \max_{a \in A} \hat{H}(a)$

car on effectue au maximum $|A|$ itération et qu'a chaque itération soit on ajoute un lassos de H dans $\mathscr{F}$ soit on multiplie $\mathscr{F}$ par 3. Comme H est obtenu à partir de notre variante de l'algorithme de Kosaraju et Sullivan, la taille de $\mathscr{F}$ est polynomiale dans la taille de $\mathcal{S}$. Ainsi, la taille de la valuation de chaque lasso dans $\mathscr{F}$ est polynomiale dans la taille de $\mathcal{S}$.

12.3 Une borne supérieure pour le nombre de lassos élémentaires distincts

Comme l'algorithme 2 termine en moins de $|A|$ itérations, il nous fournit un multi-ensemble $\mathscr{F}$ de lassos élémentaires démarrant d'un état fixé arbitraire $\hat{q}$ avec au plus $|A|$ lassos distincts. Nous pouvons faire en sorte que le multi-ensemble $\mathscr{F}$ contient au plus p lassos distincts. Ceci repose essentiellement sur un argument analogue au théorème de Carathéodory [97, Cor. 7.7i].

Propriété 12.3.1. *Soit $\hat{H}$ une circulation non vide et $\mathscr{F} = \lambda_0 \cdot W_0 + ... + \lambda_n \cdot W_n$ un multi-ensemble fini non vide de lassos élémentaires démarrant de $\hat{q}$ tel que $cout(\mathscr{F}) = m \cdot cout(\hat{H})$ pour un certain $m \in \mathbb{N}^*$. Alors il existe un multi-ensemble fini non vide $\mathscr{F}'$ de lassos élémentaires débutent de l'état $\hat{q}$ tel que $cout(\mathscr{F}') = m' \cdot cout(\hat{H})$ pour un certain $m' \in \mathbb{N}^*$ et $\mathscr{F}'$ est constitué d'au maximum p lassos élémentaires distincts.*

Démonstration. Si $n + 1 \leq p$, l'énoncé est trivial. Nous supposerons donc $n + 1 > p$. Les $n+1$ vecteurs coûts $cout(W_i)$ sont linéairement dépendants : il existe des nombres rationnels $\mu_0, ..., \mu_n$ non tous nuls tel que $\sum_{i=0}^{n} \mu_i \cdot cout(W_i) = \vec{0}$. Notons ici que les nombres rationnels $\mu_0, ..., \mu_n$ peuvent être calculés en temps polynomial en résolvant les $n + 1$ programmes linéaires suivants

$$(P_k) \begin{cases} \sum_{i=0}^{n} \mu_i \cdot cout(W_i) = \vec{0} \\ \mu_k \geq 1 \end{cases}$$

pour $k \in [0..n]$. On peut supposer que $\mu_i \in \mathbb{Z}$ pour chaque $i \in [0..n]$ et $\mu_j \geq 1$ pour chaque $j \in [0..n]$. Nous avons simplement besoin ici d'obtenir une solution entière à partir d'une solution rationnelle. Pour ce faire, nous pouvons utiliser l'algorithme d'Euclide afin de calculer le plus petit commun multiple m des dénominateurs de tous les μ_i.

Rappelons que $\lambda_k \geq 1$ pour chaque $i \in [0..n]$. Soit k tel que

$$\frac{\mu_k}{\lambda_k} = \max_{i \in [0..n]} \frac{\mu_i}{\lambda_i}$$

Alors $\mu_k > 0$, car $\mu_j \geq 1$ pour un certain $j \in [0..n]$. De plus, $\lambda_i.\mu_k - \lambda_k.\mu_i \geq 0$ pour chaque $i \in [0..n]$. Il s'ensuit que nous avons $\sum_{i=0}^{n} (\lambda_i.\mu_k - \lambda_k.\mu_i) \cdot cout(W_i) = \mu_k \cdot m \cdot cout(\hat{H})$. Comme $\lambda_k.\mu_k - \lambda_k.\mu_k = 0$ nous obtenons que le multi-ensemble de lassos élémentaires $\mathscr{F}' = \sum_{i \neq k} (\lambda_i.\mu_k - \lambda_k.\mu_i) \cdot W_i$ se compose d'au plus n lassos élémentaires et satisfait $cout(\mathscr{F}') = \mu_k \cdot cout(\hat{H})$. De plus, les lassos de $\mathscr{F}'$ sont choisis parmi les lassos de $\mathscr{F}$ et donc débutent de l'état $\hat{q}$. □

Théorème 12.3.2. *Soit $\hat{H}$ une circulation non vide et $\hat{q} \in Q_{\hat{H}}$. Nous pouvons calculer en temps polynomial un multi-ensemble $\mathscr{F}$ constitué d'au plus p lassos élémentaires distincts démarrant de l'état $\hat{q}$ tel que $cout(\mathscr{F}) = m \cdot cout(\hat{H})$ pour un certain $m \in \mathbb{N}^*$.*

Démonstration. Par la propriété 12.2.1, nous pouvons calculer en temps polynomial un multi-ensemble fini non vide de lassos élémentaires $\mathscr{F} = \lambda_0 \cdot W_0 + ... + \lambda_n \cdot W_n$ démarrant de $\hat{q}$ tel que $cout(\mathscr{F}) = m \cdot cout(\hat{H})$ pour un certain $m \in \mathbb{N}^*$. Par la propriété 12.3.1, on peut trouver un multi-ensemble fini non vide $\mathscr{F}'$ de lassos élémentaires débutent de l'état $\hat{q}$ tel que $cout(\mathscr{F}') = m' \cdot cout(\hat{H})$ pour un certain $m' \in \mathbb{N}^*$ et $\mathscr{F}'$ est constitué d'au maximum p lassos élémentaires distincts. □

Ainsi, nous sommes en mesure de décrire les cycles pathologiques pour la terminaison structurelle ou le caractère structurellement borné sous la forme d'un ensemble de lassos élémentaires en temps polynomial.

Chapitre

13 Recherche d'un contre-exemple minimal

La recherche de contre-exemples d'une propriété les plus courts possible est un problème classique qui permet de faciliter la phase de débogage. Ces contre-exemples sont plus faciles à comprendre, car ils se concentrent sur les causes réelles du bug [30, 69].

Nous souhaitons pouvoir trouver en temps polynomial un contre-exemple « minimal » pour les propriétés structurelles que nous étudions. Pour cela, deux approches apparaissent naturellement. La première vise à minimiser la longueur du contre-exemple. Du fait, l'algorithme de Kosaraju et Sullivan produit de facto une circulation avec un support maximal, une seconde approche intéressante consiste à minimiser le rapport de la circulation ou encore la taille du support. Une autre approche serait de minimiser le nombre de dimensions qui interagissent dans un cycle. Malheureusement, ces problèmes sont NP-difficiles.

Propriété 13.0.1. *Déterminer s'il existe dans un VASS donné un cycle pathologique de longueur inférieure à n est un problème NP-complet.*

Démonstration. Montrons que ce problème est dans NP. Nous pouvons utiliser comme certificat un cycle pathologique de longueur inférieure à n sous la forme de multi-ensemble d'arcs. En appliquant l'algorithme de Kosaraju et Sullivan, nous pouvons vérifier en temps polynomial que ce cycle est pathologique.

Nous effectuons une réduction de 3-SAT. Soit ϕ une formule 3-SAT avec de c clauses $C_0, ..., C_{c-1}$ et v variables booléennes $B_0, ..., B_{v-1}$. Nous considérons le VASS $\mathcal{S}_\phi = (Q, A)$ contenant $c+v$ états $q_0 ... q_{c+v-1}$ et $3\times c+2\times v$ arcs étiquetés par des vecteurs de dimension $p = 2\times v$. Chaque dimension d'un vecteur représente un littéral. Pour chaque clause C_i et chaque littéral L dans C_i, nous mettons un arc clause allant de q_i vers q_{i+1} étiqueté par un vecteur dont les coefficients sont -1 dans la dimension L et 0 dans les autres dimensions. Pour chaque variable B_j, nous mettons deux arcs littéraux allant de q_{c+j} vers $q_{c+j+1(mod\ c+v)}$. Le premier arc est étiqueté par un vecteur avec un coefficient $4c$ dans la dimension correspondant au littéral positif B_j et 0 dans les autres dimensions. Le second arc est étiqueté par un vecteur avec un coefficient $4c$ dans la dimension correspondant au littéral négatif $\neg B_j$ et 0 dans les autres dimensions. Remarquons que les deux derniers arcs clauses relient l'état q_{c+v-1} à l'état q_0.

Cette construction est illustrée par la figure 13.1. Par souci de lisibilité les arcs étiquetés par $-a$ correspondent au vecteur ayant -1 dans la dimension correspondant au littéral a et 0 dans les autres dimensions ; les arcs étiquetés par $4c \cdot a$ correspondent au vecteur avec le coefficient $4c$ dans la dimension correspondant au littéral a et 0 dans les autres dimensions.

Remarquons que le VASS $\mathcal{S}_\phi$ n'est pas structurellement borné ni structurellement terminant : le coût d'un cycle de longueur $2 \times (c + v)$ passant une fois par chaque arc littéral est strictement positif. De plus, la longueur de chaque cycle est d'au moins $c + n$. On peut

vérifier que la formule ϕ est satisfiable si, et seulement si, le VASS $\mathcal{S}_\phi$ admet un cycle γ de longueur $c + v$ avec $cout(\gamma) \geq 0$ (respectivement $cout(\gamma) \gneq 0$).

Supposons d'abord qu'il existe un cycle γ de longueur $c+v$ et avec $cout(\gamma) \geq 0$ (respectivement $cout(\gamma) \gneq 0$). Alors γ est un cycle élémentaire qui contient exactement un arc littéral de chaque variable et exactement un arc clause de chaque clause. Comme $cout(\gamma) \geq 0$, les arcs littéraux de γ correspondent à une affectation satisfaisant ϕ. Inversement, une affectation satisfaisant ϕ détermine un arc littéral pour chaque variable et un arc clause pour chaque clause. Le cycle élémentaire résultant γ satisfait $cout(\gamma) \gneq 0$ et est de longueur $c + v$. □

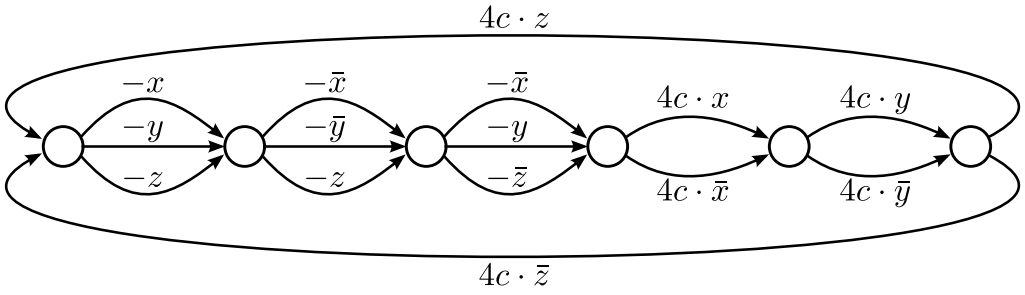


FIGURE 13.1 – Exemple de la réduction pour $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z)$.

Corollaire 13.0.2. *Trouver dans un VASS donné un cycle pathologique de longueur minimale est un problème NP-difficile.*

Corollaire 13.0.3. *Trouver dans un VASS donné un cycle pathologique avec un nombre minimal d'arcs distincts est un problème NP-difficile.*

Démonstration. Nous effectuons la même réduction de 3-SAT que pour la propriété 13.0.1.

Supposons d'abord qu'il existe un cycle γ avec $c + v$ arcs distincts et $cout(\gamma) \geq 0$ (respectivement $cout(\gamma) \gneq 0$). Alors γ est un cycle élémentaire qui contient exactement un arc littéral de chaque variable et exactement un arc clause de chaque clause. Comme $cout(\gamma) \geq 0$, les arcs littéraux de γ correspondent à une affectation satisfaisant ϕ. Inversement, une affectation satisfaisant ϕ détermine un arc littéral pour chaque variable et un arc clause pour chaque clause. Le cycle élémentaire résultant γ satisfait $cout(\gamma) \gneq 0$ et utilise $c + v$ arcs distincts. □

Une dimension $i \in [1..p]$ est dite impliquée dans un arc a si $cost(a)[i] \neq 0$. L'ensemble des dimensions en interaction dans un cycle rassemble toutes les dimensions impliquées dans ses arcs. Une autre approche consiste à minimiser le nombre de dimensions qui interagissent dans un cycle. Encore une fois, ce problème est NP-difficile.

Propriété 13.0.4. *Trouver dans un VASS donné un cycle pathologique avec au plus k dimensions en interaction est NP-complet.*

Démonstration. Montrons que ce problème est dans NP. Nous pouvons utiliser comme certificat un ensemble d'arcs interagissant avec k dimensions. En considérant seulement les arcs présents dans ce support et en appliquant l'algorithme de Kosaraju et Sullivan, nous pouvons vérifier en temps polynomial qu'il existe un cycle pathologique construit sur ce support.

Pour montrer la NP-difficulté, nous présentons une réduction de 3-SAT. Soit ϕ une formule 3-SAT de c clauses et v variables. Nous construisons un VASS $\mathcal{S}_\phi$ contenant $c+v$ états notés $C_0...C_{c+v-1}$. Les arcs du VASS sont étiquetés par des vecteurs de dimension $2\times v$. Chaque place est identifiée par une variable ou la négation d'une variable. Plus précisément, la place $2\times i$ représente la variable v_i et la place $2\times i+1$ représente sa négation.

Les c premiers états $C_0...C_{c-1}$ représentent les c clauses de ϕ. Chacun de ces états est relié au suivant par trois arcs. Plus précisément, pour chaque clause C_i avec $0 \leq i \leq c-1$, trois arcs vont de l'état C_i à l'état C_{i+1}. Chacun de ces arcs représente un littéral de la clause C_i et porte un vecteur dont les coefficients sont 1 pour la dimension du littéral correspondant et 0 pour les autres places.

Ensuite, pour chaque i avec $0 \leq i \leq v-1$, nous relions l'état C_{c+i} à l'état $C_{c+i+1(mod\ c+v)}$ avec deux arcs. Le premier arc est étiqueté par un vecteur dont les coefficients sont égaux à 1 dans la place $2\times i$ et 0 dans les autres places. Le second arc est étiqueté par un vecteur dont les coefficients sont égaux à 1 dans la place $2\times i+1$ et 0 dans les autres places. Remarquons que les deux derniers arcs clauses relient l'état q_{c+v-1} à l'état q_0. De plus, tous les coefficients sont positifs.

Cette construction est illustrée par la figure 13.2. Par souci de lisibilité nous étiquetons les arcs par le littéral dans lequel la place contient un 1.

On peut vérifier que la formule booléenne ϕ est satisfiable si, et seulement si, le VASS $\mathcal{S}_\phi$ admet un cycle pathologique impliquant au plus v dimensions. Notons tout d'abord que tout cycle implique au moins v dimensions.

— Nous supposons d'abord qu'il existe un cycle pathologique γ impliquant v places. Nous prouvons que ϕ est satisfiable. Notons que γ a besoin de parcourir tous les états de $\mathcal{S}_\phi$ et en particulier les états $C_c...C_{c+v-1}$. Au moins v places sont impliquées par les arcs de γ entre ces états. Comme seulement v places sont impliquées dans γ, il existe un chemin entre C_0 et C_{c-1} impliquant seulement des dimensions correspondant à une assignation. Ainsi, ϕ est satisfiable.

— Nous supposons maintenant que ϕ soit satisfiable. Nous considérons une affectation qui satisfait ϕ. Pour chaque variable v_i, cette assignation nous permet de choisir un arc allant de l'état C_{c+i} vers l'état $C_{c+i+1(mod\ c+v)}$. Comme chaque clause est satisfaite, nous pouvons également choisir un arc entre chaque état C_j et C_{j+1}. Comme tous les coefficients sont positifs, ce cycle est pathologique (pour la terminaison structurelle et le caractère structurellement borné).

□

Une dernière approche vise à minimiser la valeur maximale de chaque dimension dans le bilan d'un cycle pathologique. Malheureusement, ce problème est également difficile.

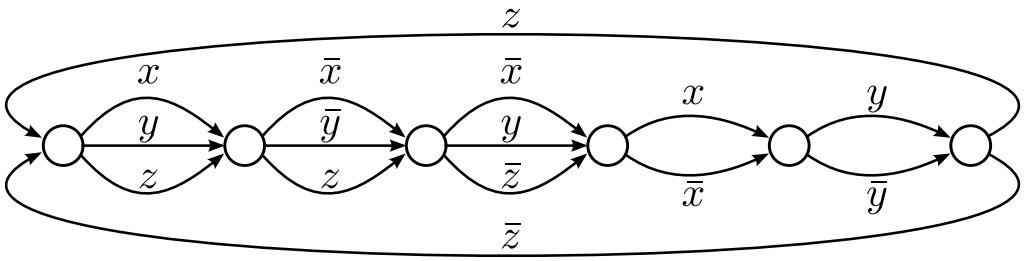


FIGURE 13.2 – Exemple de la réduction pour $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z)$.

Propriété 13.0.5. *Trouvez dans un VASS donné un cycle pathologique sous la forme d'un multi-ensemble connexe et Eulérien W minimisant $\sum_{a \in A} W[a](i)$ pour chaque place i est NP-difficile.*

Démonstration. Nous effectuons une réduction du problème 3-SAT vers ce problème. La réduction est très proche de la réduction précédente, la disposition des états et des arcs reste la même, seules les étiquettes sur arcs changent et une dimension w supplémentaire est ajoutée. Pour tous i tel que $0 \leq i < c$, les coefficients des vecteurs reliant l'état C_i a l'état C_{i+1} sont -1 dans la dimension du littéral qu'il représente, et 0 dans les autres dimensions. Pour tous i tel que $0 \leq i < v$, les coefficients des vecteurs reliant l'état C_{c+i} a l'état C_{c+i+1} sont c dans la dimension $2 \times i$ et 0 dans les autres dimensions pour le premier arc, et c dans la dimension $2 \times i + 1$ et 0 dans les autres dimensions pour le second arc. Pour terminer, les coefficients du vecteur reliant l'état C_{c+v} à l'état C_0 contient c dans la nouvelle dimension w et 0 dans les autres dimensions. Cette construction est illustrée par la figure 13.3. Par souci de lisibilité, les arcs étiquetés par $c \cdot a$ correspondent au vecteur ayant c dans la dimension correspondant au littéral a et 0 dans les autres dimensions.

Montrons que la formule F est satisfiable si et seulement si le VASS S contient un cycle pathologique W tel que $\forall i, W[i] \leq k$ avec $k = c$.

Supposons qu'il existe un cycle pathologique W tel que $W[i] \leq c$ pour chaque place i et montrons alors que la formule F est satisfiable. Ce cycle W est élémentaire, car si le dernier arc reliant l'état C_{c+v} à l'état C_0 est sélectionné deux fois, alors la valeur sur la dimension w du bilan de ce cycle dépasserait c. Comme W est un cycle élémentaire, le parcourt des états allant de C_c à C_{c+v} fait que pour chaque variable nous ajoutons $+c$ soit dans la dimension du littéral positif soit dans la dimension du littéral négatif. Comme W est également pathologique, il existe un chemin entre C_0 et C_{c-1} impliquant seulement les dimensions dont nous avons ajouté $+c$ qui correspondent à une assignation. Ainsi, F est satisfiable.

Inversement, supposons qu'il existe une assignation valide pour la formule F, montrons qu'il existe un cycle pathologique W tel que $W[i] \leq c$ pour chaque place i. Considérons un cycle élémentaire W passant seulement par des arcs impliquant les littéraux assigné à vrai dans F ou impliquant la dimension w. Comme la formule est satisfiable, un tel cycle existe. Remarquons que ce cycle est pathologique, car nous ne pouvons pas impliquer plus de c fois une même dimension entre C_0 et C_{c-1} or nous ajoutons $+c$ dans chaque

dimension correspondant à l'assignation satisfaisant F. De plus, avec un cycle élémentaire, nous ne pouvons pas dépasser la valeur c sur l'une des dimensions. Ainsi, il existe un cycle pathologique W tel que $\forall i, W[i] \leq k$. □

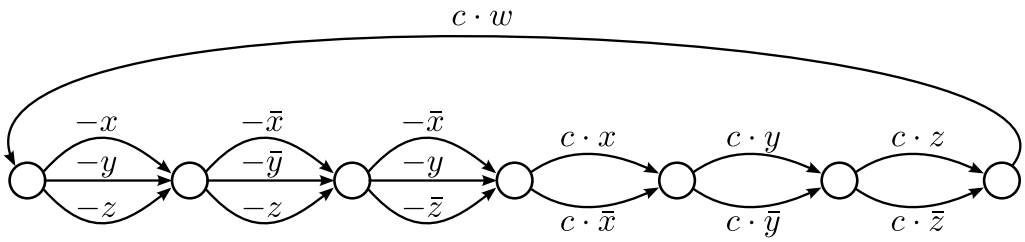


FIGURE 13.3 – Exemple de la réduction pour $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z)$.

Bien que les minimisations « naturelles » vues ci-dessus sont toutes NP-difficiles, nous souhaitons dans ce chapitre minimiser la tailles des lassos des contre-exemples en temps polynomial. Pour ce faire, nous fixons un état de départ $\hat{q}$ et un nombre naturel l et nous nous concentrons sur les lassos partant de $\hat{q}$ tels que chaque composante du lasso a une longueur d'au plus l.

13.1 Une borne supérieure pour la valuation des lassos

Nous commençons par établir cette borne pour le problème de la terminaison structurelle. Puis, nous étendrons ce résultat pour le problème du caractère structurellement borné.

13.1.1 Terminaison structurelle

Nous observons pour commencer que nous pouvons restreindre la recherche à des lassos avec une valuation au plus égal à 2^Φ où Φ est polynomial dans la taille de $\mathcal{S}$. L'exemple 13.1.1 illustre cette propriété dans le cas particulier de vecteurs de dimension 1.

Exemple 13.1.1. *Considérons le cas où $p = 1$. Supposons qu'il existe un cycle γ contenant l'état $\hat{q}$ tel que $cout(\gamma) \geq \vec{0}$. Nous distinguons deux cas. S'il existe un lasso $W = D + k \cdot C$ démarrant de $\hat{q}$ tel que $cout(C) > \vec{0}$, alors $cout(D + k' \cdot C) \geq \vec{0}$ pour un $k' \in \mathbb{N}$. De plus, nous pouvons exiger que $k' \leq 2 \times v_{max} \times Q$ avec v_{max} la valeur absolue maximale des entiers présents dans les vecteurs portés par les arcs de $\mathcal{S}$. Comme le coût de D est inférieur à $2 \times v_{max} \times Q$, s'il n'existe pas de tel lasso, alors les cycles élémentaires dans γ ont un coût ≤ 0 : nous pouvons donc extraire de γ un cycle élémentaire γ' démarrant de $\hat{q}$ tel que $cout(\gamma') \geq \vec{0}$. Ce cycle élémentaire est un lasso de valuation* 1.

Soit $\mathcal{S} = (Q, A)$ un VASS de taille $size(\mathcal{S})$, $\hat{q} \in Q$ un état fixé de $\mathcal{S}$ et $l \in \mathbb{N}$. Nous voulons savoir s'il existe un multi-ensemble $\mathscr{F}$ constitué de lassos « qui débutent de » $\hat{q}$ avec une longueur d'au plus l tel que $cout(\mathscr{F}) \geq \vec{0}$. Nous introduisons d'abord une borne supérieure pour la valuation des lassos que nous avons besoin de considérer.

Lemme 13.1.2. *Soit $\mathscr{F}$ un multi-ensemble non vide de lassos démarrant de $\hat{q}$ avec une longueur d'au plus l tel que $cout(\mathscr{F}) \geq \vec{0}$. Soit $\Phi = 96 \times p^4 \times size(\mathcal{S})$. Alors il existe un multi-ensemble fini non vide $\mathscr{F}'$ de lassos démarrant de $\hat{q}$ avec une longueur d'au plus l et une valuation d'au plus 2^Φ tel que $cout(\mathscr{F}') \geq \vec{0}$.*

Démonstration. Par le théorème 12.3.2, il existe un nombre naturel positif $n \leq p$ et n lassos $W_1, ..., W_n$ tel que le système (Sys1) de $p + n$ inégalités

$$\begin{array}{rcll} \sum_{i=1}^{n} k_i \cdot cout(W_i) & \geq & \vec{0} & \\ k_i & > & 0 & \text{pour chaque } i \in [1..n] \end{array}$$

a une solution entière. Nous posons $W_i = D_{2i} + k'_i \cdot C_{2i+1}$ avec k'_i la valuation du lasso W_i. Nous considérons maintenant le nouveau système (Sys2) de $p + 2n$ inégalités

$$\begin{array}{rcll} \sum_{i=1}^{n} k_{2i} \cdot cout(D_{2i}) + k_{2i+1} \cdot cout(C_{2i+1}) & \geq & \vec{0} & \\ k_{2i} & > & 0 & \text{pour chaque } i \in [1..n] \\ k_{2i+1} & \geq & 0 & \text{pour chaque } i \in [1..n] \end{array}$$

Comme (Sys1) a une solution entière (Sys2) a également une solution entière. Toute solution entière de (Sys2) correspond à un multi-ensemble $\mathscr{F}$ de lassos démarrant de $\hat{q}$ tel que $cout(\mathscr{F}) \geq \vec{0}$ et pour chaque i, le lasso $D_{2i} + k_{2i+1} \cdot C$ apparaît une fois et le lasso D_{2i} avec une valuation 0 apparaît $k_{2i+1} - 1$ fois si $k_{2i+1} \geq 1$. Rappelons que la résolution d'un système d'inégalité diophantienne est NP-complet. Il existe donc des solutions entières utilisant qu'un espace polynomial. La matrice de (Sys2) a $p + 2 \times n$ lignes et $2 \times n$ colonnes. La valeur absolue de chaque composante de la matrice est inférieure à $2 \times |Q| \times v_{max}$ avec v_{max} la valeur absolue maximale des entiers présents dans les vecteurs portés par les arcs de $\mathcal{S}$. On peut évidement supposer que $|A| \geq 1$, $|Q| \geq 1$ et $p \geq 1$. Alors,

$$\begin{array}{rcl} \text{size}(\mathcal{S}) & = & |A| \times (2 \times \lceil \log_2(|Q| + 1) \rceil + p \times (1 + \lceil \log_2(1 + v_{max}) \rceil)) \\ & \geq & 2 \times \lceil \log_2(|Q| + 1) \rceil + \lceil \log_2(1 + v_{max}) \rceil + 1 \\ & \geq & 2 \times \log_2(|Q| + 1) + \log_2(1 + v_{max}) + 1 \\ & \geq & \log_2(2 \times |Q| + 1) + \log_2(1 + v_{max}) + 1 \\ & \geq & \log_2((2 \times |Q| + 1) \times (1 + v_{max})) + 1 \\ & \geq & \log_2(2 \times |Q| \times v_{max} + 1) + 1 \\ & \geq & \lceil \log_2(2 \times |Q| \times v_{max} + 1) \rceil \end{array}$$

La taille de chaque ligne est de $2 \times n \times \lceil \log_2(2 \times |Q| \times v_{max} + 1) \rceil \leq 2 \times p \times size(\mathcal{S})$. Par [97, Cor.17.1b], il existe des solutions entières au système (Sys2) dont la taille est inférieure à $6 \times (2 \times p)^3 \times \varphi$, où la complexité des faces φ est plus petite que $2 \times p \times size(\mathcal{S})$. Il existe donc une solution au système (Sys2) dont la taille est inférieure à $96 \times p^4 \times size(\mathcal{S}) = \Phi$. Par conséquent, il existe une solution entière au système (Sys2) où chaque variable k_i satisfait $k_i \leq 2^\Phi$. □

Notons ici que le nombre N de lassos débutant de $\hat{q}$ avec une longueur d'au plus l et une valuation d'au plus 2^Φ est exponentielle dans la taille de $\mathcal{S}$. Soit $W_1, ..., W_N$ une énumération

de ces lassos. Alors le programme linéaire $\sum_{i=1}^{i=N} x[i] \cdot cout(W_i) \geq \vec{0}$ avec $x \in \mathbb{Q}^N$ et $x \gneq \vec{0}$ a une solution si, et seulement si, il existe un multi-ensemble non vide $\mathscr{F}$ de lassos démarrant de $\hat{q}$ avec une longueur d'au plus l (et une valuation d'au plus 2^Φ) tel que $cout(\mathscr{F}) \geq \vec{0}$. Ceci est similaire à la technique utilisée dans [100, Th. 2] pour le cas particulier des réseaux de Petri (c'est-à-dire un VASS avec un seul état) où le problème est de calculer un multi-ensemble d'arcs pour lequel le coût est positif (théorème 10.2.1).

Nous considérons en fait le problème dual. Nous définissons le programme linéaire $PL_{\mathcal{S},\hat{q},l}$ pour le vecteur $w \in \mathbb{Q}^p$ de p inconnues, avec les contraintes :

- $w[i] > 0$, pour chaque $i \in [1..p]$;
- $-cout(W)^\top w > 0$, pour chaque lasso W démarrant de $\hat{q}$ avec une longueur d'au plus l et une valuation d'au plus 2^Φ.

Le nombre de contraintes vaut $m = N + p$. Par le théorème de Gordan [97, p. 95] (voir aussi [34, Th. 2.13]), le programme linéaire $PL_{\mathcal{S},\hat{q},l}$ n'a pas de solution si, et seulement si, il existe une combinaison linéaire à coefficients positifs non tous nuls, dans la liste des lignes du système, qui est plus grand ou égal à $\vec{0}$. Ceci nous conduit à l'énoncé suivant.

Corollaire 13.1.3. *Le programme linéaire $PL_{\mathcal{S},\hat{q},l}$, n'a pas de solution si, et seulement si, il existe un multi-ensemble non vide $\mathscr{F}$ de lassos démarrant de $\hat{q}$ avec une longueur d'au plus l tel que $cout(\mathscr{F}) \geq \vec{0}$.*

Démonstration. Supposons que $PL_{\mathcal{S},\hat{q},l}$ a une solution rationnelle $w' \in \mathbb{Q}^p$. Alors, pour chaque lasso W démarrant de $\hat{q}$ avec une longueur d'au plus l et avec une valuation d'au plus 2^Φ, nous avons $cout(W)^\top w' < 0$. Soit $\mathscr{F}$ un multi-ensemble non vide de lassos démarrant de $\hat{q}$ avec une longueur d'au plus l. Supposons que $cout(\mathscr{F}) \geq \vec{0}$. Par le lemme 13.1.2, il existe un multi-ensemble non vide $\mathscr{F}'$ de lassos démarrant de $\hat{q}$ avec une longueur d'au plus l et une valuation d'au plus 2^Φ tel que $cout(\mathscr{F}') \geq \vec{0}$. Alors $cout(\mathscr{F}')^\top w' < 0$. Comme $w'[i] > 0$ pour chaque i, au moins un coefficient du vecteur $cout(\mathscr{F}')$ est strictement négatif. Contradiction.

Supposons maintenant que le système $PL_{\mathcal{S},\hat{q},l} : Aw > \vec{0}$ n'a pas de solution rationnelle. Par le théorème de Gordan, il existe une combinaison linéaire positive non nulle d'inégalités de $PL_{\mathcal{S},\hat{q},l}$ égale au vecteur nul : $y^\top A = \vec{0}$ avec $y \in \mathbb{Q}^m$ et $y \gneq \vec{0}$. On peut supposer que cette combinaison linéaire est entière, c'est-à-dire $y \in \mathbb{N}^m$. Par conséquent, il existe des lassos $W_1, ..., W_k$ démarrant de $\hat{q}$ avec une longueur maximale l, une valuation maximale 2^Φ et des poids $\lambda_1, ..., \lambda_k \in \mathbb{N}^*$ tel que $\sum_{i=1}^{i=k} \lambda_i \cdot cout(W_i) \geq \vec{0}$ avec $k \geq 1$. □

13.1.2 Caractère structurellement borné

Nous allons adapter le résultat précédant au problème du caractère structurellement borné. En effet, nous cherchons maintenant à savoir s'il existe un multi-ensemble $\mathscr{F}$ de lassos débutant de $\hat{q}$ avec une longueur d'au plus l tel que $cout(\mathscr{F}) \gneq \vec{0}$.

Lemme 13.1.4. *Soit $\mathscr{F}$ un multi-ensemble non vide de lassos démarrant de $\hat{q}$ avec une longueur d'au plus l tel que $cout(\mathscr{F}) \gneq \vec{0}$. Soit $\Phi = 96 \times p^4 \times size(\mathcal{S})$. Alors il existe un*

multi-ensemble fini non vide $\mathscr{F}'$ de lassos démarrant de $\hat{q}$ avec une longueur d'au plus l et une valuation d'au plus 2^Φ tel que $cout(\mathscr{F}') \gneq \vec{0}$.

Démonstration. La preuve est la même que pour le lemme 13.1.2 en changeant les inégalités. □

Nous définissons le programme linéaire $PL'_{\mathcal{S},\hat{q},l}$ pour le vecteur $w \in \mathbb{Q}^p$ de p inconnues, avec les contraintes :

- $w[i] > 0$, pour chaque $i \in [1..p]$;
- $-cout(W)^\top w \geq 0$, pour chaque lasso W démarrant de $\hat{q}$ avec une longueur d'au plus l et une valuation d'au plus 2^Φ.

Corollaire 13.1.5. *Le programme linéaire $PL'_{\mathcal{S},\hat{q},l}$, n'a pas de solution si, et seulement si, il existe un multi-ensemble non vide $\mathscr{F}$ de lassos démarrant de $\hat{q}$ avec une longueur d'au plus l tel que $cout(\mathscr{F}) \gneq \vec{0}$.*

Démonstration. Supposons que $PL'_{\mathcal{S},\hat{q},l}$ a une solution rationnelle $w' \in \mathbb{Q}^P$. Pour chaque lasso W démarrant de $\hat{q}$ avec une longueur d'au plus l et avec une valuation d'au plus 2^Φ, nous avons $cout(W)^\top w' \leq 0$. Soit $\mathscr{F}$ un multi-ensemble non vide de lassos démarrant de $\hat{q}$ avec une longueur d'au plus l. Supposons que $cout(\mathscr{F}) \gneq \vec{0}$. Par le lemme 13.1.4, il existe un multi-ensemble non vide $\mathscr{F}'$ de lassos démarrant de $\hat{q}$ avec une longueur d'au plus l et une valuation d'au plus 2^Φ tel que $cout(\mathscr{F}') \gneq \vec{0}$. Comme $w'[i] > 0$ pour chaque i, alors $cout(\mathscr{F}')^\top w' > 0$. Ceci contredit le fait que $cout(W)^\top w' \leq 0$ pour chaque W de $\mathscr{F}'$.

Supposons maintenant que le système $PL'_{\mathcal{S},\hat{q},l} : Aw \geq \vec{0}$ n'a pas de solution rationnelle. Par le Lemme de Farkas [46], il existe une combinaison linéaire positive d'inégalités dans $PL'_{\mathcal{S},\hat{q},l}$ avec un coût $\gneq \vec{0}$ c'est-à-dire $y^\top A \gneq \vec{0}$ et $y \geq \vec{0}$ pour $y \in \mathbb{Q}^m$. Nous pouvons supposer que cette combinaison linéaire est entière, c'est-à-dire $y \in \mathbb{N}^m$. Il y a donc des lassos $W_1, ..., W_k$ débutants de $\hat{q}$ avec une longueur maximale l, une valuation maximale 2^Φ et des poids $\lambda_1, ..., \lambda_k \in \mathbb{N}^*$ tel que $\sum_{i=1}^{i=k} \lambda_i \cdot cout(W_i) \gneq \vec{0}$ avec $k \geq 1$. □

Nous continuons à étudier par la suite seulement le problème de la terminaison structurelle. En effet, les arguments pour le caractère structurellement borné sont identiques.

13.2 Calculer un multi-ensemble pathologique de lassos de valuation maximale l

Le programme linéaire $PL_{\mathcal{S},\hat{q},l}$ contient un nombre exponentiel d'inégalités. Cependant, d'après le résultat fondamental de Grötschel, Lovász et Schrijver [97, Th. 14.1], il suffit de concevoir un algorithme de séparation polynomial pour résoudre ce problème de programmation linéaire en temps polynomial. Étant donné un vecteur $w > \vec{0}$, l'oracle de séparation doit décider si w est une solution de $PL_{\mathcal{S},\hat{q},l}$, et fournir une contrainte violée si w n'est pas une solution. D'une certaine façon, l'algorithme de séparation doit trouver un lasso W de longueur maximale l et de valuation maximale 2^Φ pour lequel $cout(W)^\top w \geq 0$ à chaque fois que w n'est pas une solution de $PL_{\mathcal{S},\hat{q},l}$. Comme expliqué dans la section ci-dessous, l'algorithme 3 est un tel oracle. Cela a pour conséquence ce résultat :

Algorithme 3 (Algorithme de séparation pour la terminaison structurelle)

Données : $\mathcal{S} = (Q, A)$ un VASS, $w \in \mathbb{Q}^p$, $\hat{q} \in Q$.
Résultat : retourne `vrai` si w est une solution de $PL_{\mathcal{S},\hat{q},l}$ et une inégalité violée sinon.

```
si w ≯ 0⃗ alors
  renvoyer un i ∈ [1..p] tel que w[i] ≤ 0.
fin si
pour q, q' ∈ Q faire
  Calculer blmw_{q,q'}(w) ∈ ℚ et un chemin σ_{q,q'} ∈ A*r en temps polynomial
fin pour
pour q ∈ Q faire
  si (*) blmw_{q̂,q}(w) + 2^Φ × blmw_{q,q}(w) + blmw_{q,q̂}(w) ≥ 0 alors
    renvoyer le vecteur ligne cout(σ_{q̂,q}) + 2^Φ × cout(σ_{q,q}) + cout(σ_{q,q̂})
  fin si
fin pour
renvoyer vrai
```

Théorème 13.2.1. *Soit $\mathcal{S} = (Q, A)$ un VASS, $\hat{q} \in Q$ un état de $\mathcal{S}$ et l un nombre naturel. Nous pouvons vérifier en temps polynomial s' il existe un multi-ensemble non vide $\mathscr{F}$ de lassos démarrant de $\hat{q}$ avec une longueur maximale l telle que $cout(\mathscr{F}) \geq \vec{0}$.*

Sans surprise, l'algorithme de Grötschel, Lovász et Schrijver pour prouver [97, Th. 14.1] peut nous fournir un certificat attestant que $PL_{\mathcal{S},\hat{q},l}$ n'a pas de solution sous la forme d'un nombre polynomial de contraintes de $PL_{\mathcal{S},\hat{q},l}$ n'ayant pas de solution. Par le théorème de Gordan, nous pouvons tirer de ce certificat un multi-ensemble $\mathscr{F}$ de lassos avec $cout(\mathscr{F}) \geq \vec{0}$. Par conséquent, nous pouvons trouver en temps polynomial un multi-ensemble de lassos avec une taille minimale qui décrit un cycle pathologique pour le problème de la terminaison structurelle. De plus, nous pouvons garantir que ce multi-ensemble contient au plus p lassos distincts (en appliquent le théorème 12.3.2).

13.3 Séparation des solutions

Soit $w \in \mathbb{Q}^p$. Nous montrons ici comment décider si w est une solution du programme linéaire $PL_{\mathcal{S},\hat{q},l}$, et comment calculer une inégalité violée sinon.

Si un composant $w[i]$ de w est négatif, alors la contrainte $w[i] > 0$ n'est pas satisfaite. On peut donc supposer que $w > \vec{0}$. Nous désignons par $\mathcal{S}/w = (Q, A/w)$ le graphe dirigé obtenu à partir du VASS $\mathcal{S}$ en remplaçant l'étiquette $cout(a) \in \mathbb{Z}^p$ de chaque arc $a \in A$ par $cout(a)^\top w$. Pour chaque paire d'états $q, q' \in Q$, nous calculons le poids maximal $\mathrm{blmw}_{q,q'}(w) \in \mathbb{Q}$ des chemins allant de q vers q' dans $\mathcal{S}/w$ avec une longueur d'au plus l. Nous calculons également le chemin $\sigma_{q,q'} \in A^*$ allant de q vers q' avec une longueur d'au plus l et tel que $cout(\sigma_{q,q'})^\top w = \mathrm{blmw}_{q,q'}(w)$. Ceci peut être fait, par exemple, par l multiplications de matrice dans l'algèbre $(max, +)$. Notons que $\mathrm{blmw}_{q,q}(w) \geq 0$ pour chaque $q \in Q$. Soit $q \in Q$ un état de $\mathcal{S}$. Si $\mathrm{blmw}_{\hat{q},q}(w) + 2^\Phi \times \mathrm{blmw}_{q,q}(w) + \mathrm{blmw}_{q,\hat{q}}(w) \geq 0$, alors le lasso W construit à partir du chemin $\sigma_{\hat{q},q}$, suivi par 2^Φ itérations du cycle $\gamma_{q,q}$ et terminant par le chemin $\sigma_{q,\hat{q}}$ satisfait $cout(W)^\top w \geq 0$ et nous fournit une contrainte violée. Cela nous amène à l'algorithme 3.

Proposition 13.3.1. *L'algorithme 3 décide si w est une solution de $PL_{\mathcal{S},\hat{q},l}$, et fourni une contrainte violée sinon.*

Démonstration. Si l'algorithme retourne une contrainte de $PL_{\mathcal{S},\hat{q},l}$, alors cette contrainte n'est pas satisfaite par w. Supposons maintenant que l'algorithme retourne `vrai`. Alors $w > \vec{0}$. Soit $W = D + k \cdot C$ un lasso démarrant de $\hat{q}$ avec une longueur d'au plus l et une valuation $k \leq 2^{\Phi}$ constituée de trois chemins σ_1, σ_2 et σ_0. Alors

$$cout(W) = cout(\sigma_1) + k \cdot cout(\sigma_0) + cout(\sigma_2)$$

$$cout(W)^{\top} w \leq \mathrm{blmw}_{\hat{q},q}(w) + k \times \mathrm{blmw}_{q,q}(w) + \mathrm{blmw}_{q,\hat{q}}(w)$$

$$cout(W)^{\top} w \leq \mathrm{blmw}_{\hat{q},q}(w) + 2^{\Phi} \times \mathrm{blmw}_{q,q}(w) + \mathrm{blmw}_{q,\hat{q}}(w)$$

car $\mathrm{blmw}_{q,q}(w) \geq 0$. Par conséquent, $cout(W)^{\top} w < 0$ et w est une solution de $\mathrm{LP}_{\mathcal{S},\hat{q},l}$. □

Conclusions et perspectives

La première partie de cette thèse est consacrée aux MSG. Nous avons étudié pour ce modèle les propriétés de divergence et de coopération globale. Afin de vérifier ces propriétés efficacement, nous les avons formalisées sous la forme de formules booléennes. Cela nous permet de tirer avantage de tous les travaux effectués sur les SAT-solveurs. Nous avons pu mesurer l'efficacité de notre approche par rapport aux outils existants sur plusieurs exemples et observer que notre approche est beaucoup plus rapide.
Nous avons également étudié la vérification des propriétés d'accessibilité. Par opposition à la divergence et la coopération globale, qui reposent sur l'analyse de tous les cycles élémentaires, vérifier les propriétés d'accessibilité nécessite de considérer tous les chemins jusqu'à une certaine longueur que nous avons déterminée. Ces problèmes sont NP-complets et peuvent également être résolus en utilisant des SAT-solveurs.

Dans la seconde partie, nous avons introduit les PNS comme une généralisation des MSG. Ce modèle peut aussi être considéré comme une généralisation des réseaux de Petri. Plus précisément, nous avons appliqué aux PNS une sémantique d'ordre partiel qui étend la sémantique de processus des réseaux de Petri. Ce modèle est non seulement plus expressif que les MSG, mais il permet également des spécifications plus concises grâce à l'ajout de variables.
Nous nous sommes intéressés à trois problèmes de vérification classiques sur l'ensemble des marquages accessibles par les préfixes des processus : le caractère borné, la couverture et l'accessibilité. Nous avons montré comment réduire ces problèmes au cas particulier des réseaux de Petri. Ainsi, tous les résultats de complexité et de décidabilité s'étendent des réseaux de Petri aux PNS sous la sémantique des processus.
Nous avons également étudié le problème consistant de vérifier si tous les processus d'un réseau de Petri avec états $\mathcal{S}$ satisfont une formule ψ exprimée en logique monadique du second ordre (MSO). Nous avons présenté une technique permettant d'établir que ce problème est décidable. Ce résultat généralise l'étude du model-checking de formules MSO sur les MSG.

La notion de non-divergence pour un MSG coïncide avec le caractère borné par préfixe pour les PNS. Cependant, l'expressivité des PNS est plus grande que celle des MSG. En particulier, vérifier la non-divergence d'un MSG est coNP-complet alors que le caractère borné par préfixe d'un PNS requiert un espace exponentiel. Pour contourner ce problème, nous proposons dans la troisième partie de la thèse d'aborder les propriétés de manière structurelle, c'est-à-dire de vérifier si, quel que soit le marquage initial, la propriété reste satisfaite. Nous montrons que la complexité des propriétés structurelles pour les PNS est polynomiale.

Nous avons également introduit la notion de propriété semi-structurelle afin de considérer des PNS paramétrés. Cela consiste à fixer le marquage initial d'un sous-ensemble approprié de places, puis à vérifier les propriétés structurelles sur les places restantes. Nous pouvons ainsi vérifier une propriété sans fixer la valeur initiale de certaines variables.

Lorsqu'une propriété structurelle ou semi-structurelle n'est pas vérifiée, il est alors intéressant d'en connaître la raison. Une manière classique est de fournir un contre-exemple à cette propriété structurelle. Comme la longueur minimale d'une séquence de règles ne respectant pas une propriété structurelle peut être de longueur exponentielle, nous avons cherché une représentation simple et compacte des contre-exemples. Nous avons alors introduit les lassos. Nous avons montré comment passer d'un contre-exemple sous la forme d'un multi-ensemble d'arcs à un contre-exemple sous la forme d'un multi-ensemble de d lassos élémentaires (d étant le nombre type de places du PNS considéré). Enfin, nous avons montré que la recherche d'un contre-exemple sous la forme d'un multi-ensemble de lassos de longueur minimale est aussi polynomiale.

Cette thèse ouvre sur plusieurs perspectives.

Perspective 1. Lorsqu'une propriété structurelle ou semi-structurelle n'est pas satisfaite, il peut être intéressant en pratique de disposer d'un contre-exemple de longueur minimale ou utilisant un nombre minimal de dimensions. Nous avons montré que ces problèmes sont NP-difficiles. En suivant l'approche de la première partie, il serait intéressant en pratique de chercher à effectuer une réduction vers SAT ou bien des solveurs de programmation linéaire en nombres entiers afin de résoudre efficacement ces problèmes.

Perspective 2. La notion de non-divergence pour un MSG coïncide avec le caractère borné par préfixe pour les PNS. Cependant, l'expressivité des PNS est plus grande que celle des MSG. En particulier, vérifier la non-divergence d'un MSG est coNP-complet alors que le caractère borné par préfixe d'un PNS requiert un espace exponentiel. Il est sûrement possible de contourner ce problème pour la classe des MSG étendus. L'idée est de déplier le MSG étendu considéré afin d'effacer toute trace de variable, puis d'appliquer les réductions vers SAT développées dans la première partie.

Perspective 3. Nous avons développé un prototype permettant de spécifier des PNS puis de vérifier certaines propriétés. Ce prototype pourrait être amélioré afin de donner lieu à un outil complet. La prise en charge des MSG étendus ainsi que la recherche d'un contre exemple minimal sont des pistes d'amélioration possible.

Perspective 4. Nous avons présenté une réduction vers SAT afin de vérifier l'accessibilité et la couverture des préfixes pour un MSG non divergent. Nous n'avons cependant pas implémenté ces formules. Il serait donc utile de les implémenter et de comparer l'efficacité de cette approche par rapport à des outils tels que SPIN.

Perspective 5. Nous avons observé qu'il est indécidable de déterminer si les processus d'un PNS sont inclus dans les processus d'un autre PNS, même lorsqu'il s'agit de MSG. Le problème est cependant décidable pour les MSG globalement coopératifs. Il serait intéressant de définir une notion de PNS globalement coopératifs pour lesquels le problème de l'inclusion serait décidable.

Bibliographie

[1] GLPK. `http://www.gnu.org/software/glpk/`.

[2] Minisat2. `http://minisat.se`.

[3] The MONA Project. `http://www.brics.dk/mona`. [Online ; accessed 7-Oct-2012].

[4] TIme petri Net Analyzer (TINA). `http://projects.laas.fr/tina`.

[5] Vass Checker (VaChe). `http://pageperso.lif.univ-mrs.fr/~florent.avellaneda/LeVaChe`.

[6] A Scenario Oracle and Formal Analysis Toolbox (SOFAT). `http://www.irisa.fr/distribcom/Prototypes/SOFAT`, 2012. [Online ; accessed 7-Oct-2012].

[7] D. Achlioptas. Random satisfiability. *Handbook of Satisfiability*, 185 :245, 2009.

[8] S. Akshay, B. Genest, L. Hélouët, and S. Yang. Regular set of representatives for time-constrained MSC graphs. *Inf. Process. Lett.*, 112(14-15) :592–598, 2012.

[9] R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence charts. *IEEE Transactions on Software Engineering*, 29(7) :623–633, 2003.

[10] R. Alur, K. Etessami, and M. Yannakakis. Realizability and verification of MSC graphs. *Theor. Comput. Sci.*, 331(1) :97–114, 2005.

[11] R. Alur and M. Yannakakis. Model checking of message sequence charts. In J. C. M. Baeten and S. Mauw, editors, *CONCUR*, volume 1664 of *Lecture Notes in Computer Science*, pages 114–129. Springer, 1999.

[12] Roberto M. Amadio and Denis Lugiez, editors. *CONCUR 2003 - Concurrency Theory, 14th International Conference, Marseille, France, September 3-5, 2003, Proceedings*, volume 2761 of *Lecture Notes in Computer Science*. Springer, 2003.

[13] F. Avellaneda and R. Morin. Checking non-divergence, channel-bound and global cooperation using SAT-solvers. In *Application of Concurrency to System Design (ACSD), 2011 11th International Conference on*, pages 19–28. IEEE, 2011.

[14] F. Avellaneda and R. Morin. Checking partial-order properties of vector addition systems with states. In *Application of Concurrency to System Design (ACSD), 2013 13th International Conference on*, pages 100–109. IEEE, 2013.

[15] N. Baudru and R. Morin. Synthesis of safe message-passing systems. In Vikraman Arvind and Sanjiva Prasad, editors, *FSTTCS*, volume 4855 of *Lecture Notes in Computer Science*, pages 277–289. Springer, 2007.

[16] Nicolas Baudru and Rémi Morin. The pros and cons of netcharts. In Gardner and Yoshida [47], pages 99–114.

[17] Nicolas Baudru and Rémi Morin. The synthesis problem of netcharts. In Donatelli and Thiagarajan [39], pages 84–104.

[18] H. Ben-Abdallah and S. Leue. Syntactic detection of process divergence and non-local choice in message sequence charts. In Brinksma [26], pages 259–274.

[19] H. Ben-Abdallah and S. Leue. MESA : Support for scenario-based design of concurrent systems. In Bernhard Steffen, editor, *TACAS*, volume 1384 of *Lecture Notes in Computer Science*, pages 118–135. Springer, 1998.

[20] E. Best and R. R. Devillers. Sequential and concurrent behaviour in Petri net theory. *Theoretical Computer Science*, 55(1) :87–136, 1987.

[21] M. Bezdeka, O. Bouda, L. Korenciak, M. Madzin, and V. Rehák. Sequence chart studio. In Jens Brandt and Keijo Heljanko, editors, *ACSD*, pages 148–153. IEEE, 2012.

[22] B. Bollig. *Automata and Logics for Message Sequence Charts.* Thèse de doctorat, Department of Computer Science, RWTH Aachen, Germany, May 2005.

[23] B. Bollig, C. Kern, M. Schlütter, and V. Stolz. MSCan - a tool for analyzing MSC specifications. In Holger Hermanns and Jens Palsberg, editors, *TACAS*, volume 3920 of *Lecture Notes in Computer Science*, pages 455–458. Springer, 2006.

[24] B. Bollig and M. Leucker. Message-passing automata are expressively equivalent to EMSO logic. *Theoretical Computer Science*, 358(2-3) :150–172, 2006.

[25] W. Brauer, editor. *Net Theory and Applications, Proceedings of the Advanced Course on General Net Theory of Processes and Systems, Hamburg, October 8-19, 1979*, volume 84 of *Lecture Notes in Computer Science*. Springer, 1980.

[26] Ed Brinksma, editor. *Tools and Algorithms for Construction and Analysis of Systems, Third International Workshop, TACAS '97, Enschede, The Netherlands, April 2-4, 1997, Proceedings*, volume 1217 of *Lecture Notes in Computer Science*. Springer, 1997.

[27] J.R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6 :66–92, 1960.

[28] Benoît Caillaud, Philippe Darondeau, Loïc Hélouët, and Gilles Lesventes. Hmscs as partial specifications ... with pns as completions. In Cassez et al. [29], pages 125–152.

[29] Franck Cassez, Claude Jard, Brigitte Rozoy, and Mark Dermot Ryan, editors. *Modeling and Verification of Parallel Processes, 4th Summer School, MOVEP 2000, Nantes, France, June 19-23, 2000*, volume 2067 of *Lecture Notes in Computer Science*. Springer, 2001.

[30] E. M. Clarke, O. Grumberg, K. L. McMillan, and X. Zhao. Efficient generation of counterexamples and witnesses in symbolic model checking. In *Proceedings of the 32nd annual ACM/IEEE Design Automation Conference*, DAC '95, pages 427–432, New York, NY, USA, 1995. ACM.

[31] M. Clerbout, M. Latteux, and Y. Roos. Semi-Commutations. In V. Diekert and G. Rozenberg, editors, *The Book of Traces*, chapter 12, pages 487–552. World Scientific Publ. Co., 1995.

[32] E. Cohen and N. Megiddo. Strongly polynomial-time and NC algorithms for detecting cycles in dynamic graphs (preliminary version). In David S. Johnson, editor, *STOC*, pages 523–534. ACM, 1989.

[33] B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In *Handbook of Graph Grammars*, pages 313–400, 1997.

[34] G.B. Dantzig and M.N. Thapa. *Linear Programming*. Springer, 2003.

[35] Ph. Darondeau. Unbounded Petri Net Synthesis. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 413–438. Springer, 2003.

[36] L.E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4) :413–422, 1913.

[37] V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, 1995.

[38] R. Diestel. *Graph Theory*. Springer-Verlag, Heidelberg, 2010.

[39] Susanna Donatelli and P. S. Thiagarajan, editors. *Petri Nets and Other Models of Concurrency - ICATPN 2006, 27th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, Turku, Finland, June 26-30, 2006, Proceedings*, volume 4024 of *Lecture Notes in Computer Science*. Springer, 2006.

[40] M. Droste, P. Gastin, and D. Kuske. Asynchronous cellular automata for pomsets. *Theoretical Computer Science*, 247(1-2) :1–38, 2000.

[41] A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-structures. Part II : State spaces of concurrent systems. *Acta Informatica*, 27(4) :343–368, 1989.

[42] J. Engelfriet. Branching processes of Petri nets. *Acta Informatica*, 28 :575–591, 1991.

[43] J. Esparza and M. Nielsen. Decidability issues for Petri nets - a survey. *Bulletin of the EATCS*, 52 :244–262, 1994.

[44] J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. *Formal Methods in System Design*, 20 :285–310, 2002.

[45] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8 :128–140, 1741.

[46] J. Farkas. Theorie der einfachen ungleichungen. *Journal für die reine und angewandte Mathematik*, 124 :1–27, 1902.

[47] Philippa Gardner and Nobuko Yoshida, editors. *CONCUR 2004 - Concurrency Theory, 15th International Conference, London, UK, August 31 - September 3, 2004, Proceedings*, volume 3170 of *Lecture Notes in Computer Science*. Springer, 2004.

[48] P. Gastin, K. N. Kumar, and M. Mukund. Reachability and boundedness in time-constrained MSC graphs. *Perspectives in Concurrency–A Festschrift for PS Thiagarajan. Universities Press, India*, 2009.

[49] B. Genest. *The odyssey of MSC-graphs*. Thèse de doctorat, Université Paris 7, 2004.

[50] B. Genest, D. Kuske, and A. Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inf. Comput.*, 204(6) :920–956, 2006.

[51] B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state high-level MSCs : Model-checking and realizability. *Journal of Computer and System Sciences*, 72(4) :617–647, 2006.

[52] H.J. Genrich and E. Stankiewicz-Wiechno. A dictionary of some basic notions of net theory. In Brauer [25], pages 519–531.

[53] R. J. van Glabbeek, U. Goltz, and J.-W. Schicke. On causal semantics of Petri nets. In Joost-Pieter Katoen and Barbara König, editors, *CONCUR*, volume 6901 of *Lecture Notes in Computer Science*, pages 43–59. Springer, 2011.

[54] U. Goltz and W. Reisig. The non-sequential behavior of Petri nets. *Information and Control*, 57(2/3) :125–147, 1983.

[55] J. Grabowski. On partial languages. *Fundamenta Informaticae*, 4(2) :427–498, 1981.

[56] E. L. Gunter, A. Muscholl, and D. Peled. Compositional message sequence charts. *International Journal on Software Tools for Technology Transfer*, 5(1) :78–89, 2003.

[57] J. Gutierrez and J. C. Bradfield. Model-checking games for fixpoint logics with partial order models. *Inf. Comput.*, 209(5) :766–781, 2011.

[58] L. Hélouët. *Analyse des exigences des systèmes répartis exprimées par des langages de scénarios*. PhD thesis, 2000.

[59] J. G. Henriksen, M. Mukund, K. Narayan Kumar, M. A. Sohoni, and P. S. Thiagarajan. A theory of regular MSC languages. *Information and Computation*, 202(1) :1–38, 2005.

[60] G. J. Holzmann, D. A. Peled, and M. H. Redberg. Design tools for requirements engineering. *Bell Labs Technical Journal*, 2 :86–95, 1997.

[61] P. W. Hoogers, H. C. M. Kleijn, and P. S. Thiagarajan. A trace semantics for Petri nets. *Information and Computation*, 117(1) :98–114, 1995.

[62] J.E. Hopcroft and J-J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8 :135–159, 1979.

[63] ITU. *Recommendation Z.120 : Message Sequence Chart (MSC)*. Haugen (ed.), Geneva, 1999.

[64] K. Iwano and K. Steiglitz. Testing for cycles in infinite graphs with periodic structure (extended abstract). In Alfred V. Aho, editor, *STOC*, pages 46–55. ACM, 1987.

[65] D. Kőnig. *Theorie der endlichen und unendlichen Graphen*. Leipzig : Akademische Verlagsgesellschaft, 1936.

[66] A. Kiehn. On the interrelation between synchronized and non-synchronized behaviour of Petri nets. *Journal of Information Processing and Cybernetics / Elektronische Informationsverarbeitung und Kybernetik*, 24(1/2) :3–18, 1988.

[67] S R. Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 267–281. ACM, 1982.

[68] S. R. Kosaraju and G. F. Sullivan. Detecting cycles in dynamic graphs in polynomial time (preliminary version). In Janos Simon, editor, *STOC*, pages 398–406. ACM, 1988.

[69] O. Kupferman and S. Sheinvald-Faragy. Finding shortest witnesses to the nonemptiness of automata on infinite words. In *Proceedings of the 17th international conference on Concurrency Theory*, CONCUR'06, pages 492–508, Berlin, Heidelberg, 2006. Springer-Verlag.

[70] J.-L. Lambert. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99(1) :79–104, 1992.

[71] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7) :558–565, 1978.

[72] J. Leroux. The general vector addition system reachability problem by Presburger inductive invariants. In *24th IEEE Symposium on Logic in Computer Science (LICS 2009), 11-14 August 2009, Los Angeles, California, USA, Proceedings*, pages 4–13. IEEE Computer Society, 2009.

[73] J. Leroux. Vector addition system reachability problem : A short self-contained proof. In Adrian Horia Dediu, Shunsuke Inenaga, and Carlos Martín-Vide, editors, *LATA*, volume 6638 of *Lecture Notes in Computer Science*, pages 41–64. Springer, 2011.

[74] R.J. Lipton. The reachability problem requires exponential space. Technical Report 63, Yale University, 1976.

[75] M. Lohrey and A. Muscholl. Bounded MSC communication. *Information and Computation*, 189(2) :160–181, 2004.

[76] P. Madhusudan. Reasoning about sequential and branching behaviours of message sequence graphs. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *ICALP*, volume 2076 of *Lecture Notes in Computer Science*, pages 809–820. Springer, 2001.

[77] P. Madhusudan and B. Meenakshi. Beyond message sequence graphs. In Ramesh Hariharan, Madhavan Mukund, and V. Vinay, editors, *FSTTCS*, volume 2245 of *LNCS*, pages 256–267. Springer, 2001.

[78] P. Madhusudan and Gennaro Parlato. The tree width of auxiliary storage. In Thomas Ball and Mooly Sagiv, editors, *POPL*, pages 283–294. ACM, 2011.

[79] S. Mauw, M. A. Reniers, and T. A. C. Willemse. Message sequence charts in the software engineering process. In *In Handbook of Software Engineering and Knowledge Engineering*, pages 437–464, 2001.

[80] E. W Mayr. An algorithm for the general Petri net reachability problem. *SIAM Journal on computing*, 13(3) :441–460, 1984.

[81] G. Memmi and G. Roucairol. Linear algebra in net theory. In Brauer [25], pages 213–223.

[82] R. Morin. Recognizable sets of message sequence charts. In Helmut Alt and Afonso Ferreira, editors, *STACS*, volume 2285 of *Lecture Notes in Computer Science*, pages 523–534. Springer, 2002.

[83] M. Mukund. Petri nets and step transition systems. *International Journal of Foundations of Computer Science*, 3(4) :443–478, 1992.

[84] A. Muscholl and Doron Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. In Miroslaw Kutylowski, Leszek Pacholski, and Tomasz Wierzbicki, editors, *MFCS*, volume 1672 of *Lecture Notes in Computer Science*, pages 81–91. Springer, 1999.

[85] M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, part I. *Theoretical Computer Science*, 13 :85–108, 1981.

[86] J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs, New Jersey : Prentice Hall, Inc., 1981.

[87] C. A. Petri. Kommunikation mit automaten. 1962.

[88] C. A. Petri. *Non-Sequential Processes : Translation of a Lecture given at the IMMD Jubilee Colloquium on 'Parallelism in Computer Science', Universität Erlangen-Nürnberg.* St. Augustin : Gesellschaft für Mathematik und Datenverarbeitung Bonn, Interner Bericht ISF-77–5, 1977.

[89] V. Pratt. Modelling concurrency with partial orders. *International Journal of Parallel Programming*, 15 :33–71, 1986.

[90] C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6 :223–231, 1978.

[91] W. Reisig. *Petri Nets, An Introduction.* EATCS Monographs, vol. 4. Springer, 1985.

[92] M.A. Reniers. *Message sequence chart : Syntax and semantics.* PhD thesis, Eindhoven University of Technology, 1998.

[93] C. Reutenauer. *The Mathematics of Petri Nets.* New York, USA : Prentice-Hall, 1990.

[94] G. S Sacerdote and R. L Tenney. The decidability of the reachability problem for vector addition systems (preliminary version). In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 61–76. ACM, 1977.

[95] J. Sakarovitch. The "last" decision problem for rational trace languages. In Simon [99], pages 460–473.

[96] J. Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, 2009.

[97] A. Schrijver. *Theory of linear and integer programming.* J. Wiley & Sons, Inc., New York, NY, USA, 1986.

[98] D. Seese. The structure of the models of decidable monadic theories of graphs. *Annals of pure and applied logic*, 53(2) :169–195, 1991.

[99] I. Simon, editor. *LATIN '92, 1st Latin American Symposium on Theoretical Informatics, São Paulo, Brazil, April 6-10, 1992, Proceedings*, volume 583 of *Lecture Notes in Computer Science*. Springer, 1992.

[100] D.D. Sleator. Data structures and terminating Petri nets. In Simon [99], pages 488–497.

[101] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2) :146–160, 1972.

[102] P.S. Thiagarajan. Regular event structures and finite Petri nets : a conjecture. In *Formal and natural computing*, pages 244–253. Springer, 2002.

[103] W. Vogler. *Modular Construction and Partial Order Semantics of Petri Nets*, volume 625 of *Lecture Notes in Computer Science*. Springer, 1992.

[104] W. Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O. - Informatique Théorique et Applications*, 21 :99–135, 1987.

Table des figures

Liste des symboles et notations

$CS(\mathcal{S})$ Ensemble de toutes les séquences de règles de $\mathcal{S}$.

$FCS(\mathcal{S})$ Ensemble de toutes les séquences de règles tirables de $\mathcal{S}$.

$r^\bullet$ Postset de la règle r, page 11.

${}^\bullet r$ Préset de la règle r, page 11.

A_C Ensemble des arcs du multi-ensemble d'arcs C.

A_σ Ensemble des arcs du chemin σ.

$\#^a(M)$ Nombre d'événements étiquetés par a dans M.

$cout(\gamma)$ Somme des vecteurs du cycle γ.

$\mu^\circ{}_s$ Marquage atteint par s dans $\mathcal{S}^\circ$, page 89.

μ_u Marquage atteint par u depuis μ_{in}, page 11.

$\mathrm{Pref}(\mathcal{L})$ Ensemble des préfixes de $\mathcal{L}$, page 13.

$[\![u]\!]_\mu$ Classe de tous les processus de u à partir du marquage μ, définition 7.1.1.

$[\![\![u]\!]\!]_\mu$ Classe de tous les processus de u respectant la sémantique FIFO à partir du marquage μ dans un MSG étendu.

$\mathrm{req}(u)$ Prérequis d'une séquence de règles u, définition 7.2.2.

$\mathrm{size}(\mathcal{S})$ Taille d'un PNS $\mathcal{S}$.

$\vec{x} > \vec{0}$ Chaque coefficient du vecteur $\vec{x}$ est strictement plus grand que 0.

$\vec{x} \geq \vec{0}$ Chaque coefficient du vecteur $\vec{x}$ est plus grand ou égal à 0.

$\vec{x} \gneq \vec{0}$ Chaque coefficient du vecteur $\vec{x}$ est plus grand ou égal à 0 et $\vec{x} \neq \vec{0}$.

$e \rightsquigarrow_\chi f$ Voir la page 22.

$LE(t)$ Ensemble des extensions linéaires de t.

$Q_{\hat{H}}$ Ensemble des états du graphe $\hat{H}$.

MSC Message Sequence Chart.

MSG Message Sequence Graph.

PNS Petri Net with States (en français « réseau de Petri avec états »).

VASS Vector Addition Systems with States.

Index

Résumé

Les MSG (pour « Message Sequence Graphs ») sont un formalisme bien connu et souvent utilisé pour décrire des ensembles de scénarios de manière visuelle dans le domaine des protocoles de communication. Nous nous intéressons dans la première partie de la thèse à la détection de la divergence, la vérification de la coopération globale ainsi que la vérification de propriétés d'accessibilité et de couverture. Notre première contribution consiste à utiliser des solveurs SAT afin de résoudre ces problèmes efficacement.

Afin de munir le formalisme des MSG de compteurs, de timers et d'autres aspects, nous introduisons le modèle des « réseaux de Petri avec états » et une sémantique de processus non-branchants. Ce modèle est non seulement plus expressif que les MSG, mais il permet également des spécifications plus concises. Nous nous intéressons à trois problèmes de vérification classiques sur l'ensemble des marquages accessibles par les préfixes des processus : le caractère borné, la couverture et l'accessibilité. De plus, nous montrons que toute propriété MSO peut être vérifiée sur un réseau borné.

Afin de considérer des systèmes paramétrés, nous introduisons également la notion de borne semi-structurelle. Cela consiste à fixer le marquage initial d'un sous-ensemble approprié de places, puis à vérifier que le système est borné, quelles que soient les valeurs des paramètres. Nous montrons comment un dépliage conduit à un problème plus facile à vérifier.

Une caractéristique particulièrement attrayante des réseaux de Petri avec états réside dans leur représentation graphique similaire à un automate. Il est donc intéressant de décrire les bugs structurels de manière visuelle. Nous montrons comment calculer en temps polynomial une représentation simple et concise d'un bug structurel.

Abstract

Message Sequence Charts (MSCs) are a popular model often used for the documentation of telecommunication protocols. In the first part of the thesis, we focus on detecting process divergence, checking global-cooperation and checking reachability properties. Our first contribution is to use SAT solvers to solve these problems effectively.

In order to study MSC specifications with counters, timers and other features, we introduce the model of Petri nets with states together with a non-branching non-sequential process semantics. We obtain a framework that is more expressive and more concise than MSGs. We consider then three classical verification problems for the set of markings reached by prefixes of processes : boundedness, covering and reachability.

We consider also the notion of semi-structural property in order to study parametrized systems. In this way, only part of the places are provided with an initial marking. Unfolding such a system leads to a simpler problem in the form of a linear programme. Moreover, we show that any MSO property can be verified on a bounded system.

A particularly attractive feature of MSG and PNS lies in their graphical representation similar to an automaton. So, it is interesting to describe the bugs visually. We show how to compute in polynomial time a simple and concise representation of a bug.