NNT : 2016SACLN017

Thèse de Doctorat
de
l'Université de Montréal
et de
l'Université Paris-Saclay
préparée à l'École normale supérieure de Cachan

Faculté des arts et des sciences
Département d'informatique et de recherche opérationnelle

École doctorale N°580
Sciences et technologies de l'information et de la communication

Spécialité de doctorat : informatique

Par

Michael Blondin

Algorithmique et complexité des systèmes à compteurs

Thèse soutenue à Montréal, le 29 juin 2016

Composition du jury :

Michel Boyer	Professeur, Université de Montréal	Président du jury / président-rapporteur
Alain Finkel	Professeur, ENS Cachan	Directeur de thèse / directeur de thèse
Jérôme Leroux	Directeur de recherche, CNRS	Rapporteur / membre du jury
Pierre McKenzie	Professeur, Université de Montréal	Directeur de thèse / directeur de thèse
Joël Ouaknine	Professeur, Oxford University	Rapporteur / examinateur externe
Sylvain Schmitz	Maître de conférences, ENS Cachan	Examinateur / membre du jury
Dimitris Koukoulopoulos	Professeur, Université de Montréal	Représentant de la doyenne de la FAS

RÉSUMÉ

L'un des aspects fondamentaux des systèmes informatiques modernes, et en particulier des systèmes critiques, est la possibilité d'exécuter plusieurs processus, partageant des ressources communes, de façon simultanée. De par leur nature concurrentielle, le bon fonctionnement de ces systèmes n'est assuré que lorsque leurs comportements ne dépendent pas d'un ordre d'exécution prédéterminé. En raison de cette caractéristique, il est particulièrement difficile de s'assurer qu'un système concurrent ne possède pas de faille.

Dans cette thèse, nous étudions la vérification formelle, une approche algorithmique qui vise à automatiser la vérification du bon fonctionnement de systèmes concurrents en procédant par une abstraction vers des modèles mathématiques. Nous considérons deux de ces modèles, les réseaux de Petri et les systèmes d'addition de vecteurs, et les problèmes de vérification qui leur sont associés.

Nous montrons que le problème d'accessibilité pour les systèmes d'addition de vecteurs (avec états) à deux compteurs est PSPACE-complet, c'est-à-dire complet pour la classe des problèmes solubles à l'aide d'une quantité polynomiale de mémoire. Nous établissons ainsi la complexité calculatoire précise de ce problème, répondant à une question demeurée ouverte depuis plus de trente ans.

Nous proposons une nouvelle approche au problème de couverture pour les réseaux de Petri, basée sur un algorithme arrière guidé par une caractérisation logique de l'accessibilité dans les réseaux de Petri continus. Cette approche nous a permis de mettre au point un nouvel algorithme qui s'avère particulièrement efficace en pratique, tel que démontré par notre implémentation logicielle nommée QCover.

Nous complétons ces résultats par une étude des systèmes de transitions bien structurés qui constituent une abstraction générale des systèmes d'addition de vecteurs et des réseaux de Petri. Nous considérons le cas des systèmes de transitions bien structurés à branchement infini, une classe qui inclut les réseaux de Petri possédant des arcs pouvant consommer ou produire un nombre arbitraire de jetons. Nous développons des outils mathématiques facilitant l'étude de ces systèmes et nous délimitons les frontières au-delà desquelles la décidabilité des problèmes de terminaison, de finitude, de maintenabilité et de couverture est perdue.

Mots-clés: vérification formelle, complexité du calcul, algorithmique, réseaux de Petri, systèmes d'addition de vecteurs, systèmes de transitions bien structurés, problème d'accessibilité, problème de couverture, branchement infini.

ABSTRACT

One fundamental aspect of computer systems, and in particular of critical systems, is the ability to run simultaneously many processes sharing resources. Such concurrent systems only work correctly when their behaviours are independent of any execution ordering. For this reason, it is particularly difficult to ensure the correctness of concurrent systems.

In this thesis, we study formal verification, an algorithmic approach to the verification of concurrent systems based on mathematical modeling. We consider two of the most prominent models, Petri nets and vector addition systems, and their usual verification problems considered in the literature.

We show that the reachability problem for vector addition systems (with states) restricted to two counters is PSPACE-complete, that is, it is complete for the class of problems solvable with a polynomial amount of memory. Hence, we establish the precise computational complexity of this problem, left open for more than thirty years.

We develop a new approach to the coverability problem for Petri nets which is primarily based on applying forward coverability in continuous Petri nets as a pruning criterion inside a backward coverability framework. We demonstrate the effectiveness of our approach by implementing it in a tool named QCover.

We complement these results with a study of well-structured transition systems which form a general abstraction of vector addition systems and Petri nets. We consider infinitely branching well-structured transition systems, a class that includes Petri nets with special transitions that may consume or produce arbitrarily many tokens. We develop mathematical tools in order to study these systems and we delineate the decidability frontier for the termination, boundedness, maintainability and coverability problems.

Keywords: formal verification, computational complexity, algorithmics, Petri nets, vector addition systems, well-structured transition systems, reachability problem, coverability problem, infinite branching.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ALGORITHMES

LISTE DES ABRÉVIATIONS

resp.	respectivement
t.q.	tel que / tels que / telle que / telles que

NOTATION

$\mathbb{N}$	ensemble des entiers naturels : $\{0, 1, 2, \ldots\}$
$\mathbb{N}_{>0}$	ensemble des entiers naturels non négatifs : $\{1, 2, \ldots\}$
$\mathbb{Z}$	ensemble des entiers : $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
$[a]$	ensemble des entiers de 1 à a : $\{1, 2, \ldots, a\}$
$[a, b]$	ensemble des entiers de a à d : $\{a, a+1, \ldots, b\}$
$[a, \infty)$	ensemble des entiers plus grands où égaux à a : $\{a, a+1, \ldots\}$
$\mathbb{Q}$	ensemble des nombres rationnels
$\mathbb{Q}_{\geq 0}$	ensemble des nombres rationnels non négatifs
$\mathbb{Q}_{>0}$	ensemble des nombres rationnels positifs
2^E	ensemble des parties de E : $\{A : A \subseteq E\}$
$a \bmod b$	reste de la division entière $a \div b$
$\log n$	logarithme de n en base 2

Mathematics, rightly viewed,
possesses not only truth, but supreme beauty —
a beauty cold and austere, like that of sculpture,
without appeal to any part of our weaker nature,
without the gorgeous trappings of painting or music,
yet sublimely pure, and capable of a stern perfection
such as only the greatest art can show.

Bertrand Russell

If you want more effective programmers,
you will discover that
they should not waste their time debugging,
they should not introduce the bugs to start with.

Edsger W. Dijkstra

REMERCIEMENTS

Cette thèse marque non seulement la fin de mes études doctorales, mais également l'aboutissement de l'ensemble de ma scolarité. Un grand nombre de personnes et d'institutions ont rendu possible mon parcours scolaire de la maternelle au doctorat et j'aimerais ainsi les remercier, sans pouvoir espérer toutes les nommer.

Je remercie d'abord mes parents de m'avoir soutenu durant toutes ces années et de m'avoir offert des conditions de vie sans lesquelles il m'aurait été difficile de poursuivre d'aussi longues études. Je remercie ma soeur, ma famille et mes amis d'hier et d'aujourd'hui. En particulier, j'aimerais remercier les gens que j'ai côtoyés durant mon doctorat et qui ont rendu ma vie plus agréable, dont : Karthik Aluru, Joao Flavio Barreto Fernandes, Xavier Bhérer-Simard, Marianne Borys, Fannie Bourdages, Boris Clain, Jhomary Corbeil, Gabriel de Faro, Bernat Domènech Plana, Kevince Dussault, David Fortin, Étienne Garbugli, Marikym Gaudreault, Adrianne Gauvin-Sasseville, Gabriel Ghio, Paul-Virak Khuong, Rébecca Lapointe, Vanessa Lessard, Sarah Montminy, Serge-Olivier Paquette, Roberto Passos Rocha, Devesh Pawar, Paul Raymond-Robichaud, Maxime Riendeau, Joan Rodríguez ; ainsi que Carole Dodeman et Lara Mahi pour m'avoir accueilli durant mes séjours en France. Je tiens particulièrement à remercier Férédéricke Barrette-Perreault pour sa présence et sa patience dans les moments plus difficiles.

J'aimerais exprimer ma profonde gratitude envers mes directeurs Alain Finkel et Pierre McKenzie qui m'ont accompagné étroitement dans toutes les étapes de mon cheminement et sans qui je n'aurais jamais pu rédiger cette thèse. Dans le cadre de mon doctorat, j'ai également eu la chance de travailler et d'échanger avec un certain nombre de collègues du LITQ et du LSV, dont mes coauteurs Stefan Göller, Christoph Haase et Serge Haddad envers qui je suis spécialement reconnaissant. Je remercie aussi le CNRS, le FRQNT et le LSV de m'avoir soutenu financièrement.

Finalement, je remercie mes très grands amis Michaël Cadilhac, Mathieu Janelle Gravel, Philippe Lamontagne et Simon Lamontagne pour tous les moments inoubliables passés en leur compagnie ainsi que leur soutien inconditionnel.

1

INTRODUCTION

En 1965, l'éminent informaticien et mathématicien Edsger W. Dijkstra posait la question suivante[1] à l'examen final de l'un des cours qu'il enseignait alors à l'Université technique d'Eindhoven :

> Quatre philosophes sont assis autour d'une table ronde, telle qu'illustrée à la figure 1.1, et réfléchissent sur les plus grands enjeux de leur société. Au centre de la table se trouve un grand bol de nouilles. Quatre baguettes sont réparties sur la table, l'une entre chaque paire de philosophes adjacents. Une personne peut prendre ou déposer une baguette située immédiatement à sa gauche ou à sa droite, mais pas les autres qui sont hors de portée. Lorsque l'un des philsophes a faim, il voudra éventuellement se nourrir afin de poursuivre ses réflexions. Il peut se nourrir en se servant dans le bol de nouilles, pourvu qu'il ait ses deux baguettes en main. Impatientée par l'attente ou distraite par l'appétit, une personne qui tient au moins une baguette ne peut plus réfléchir tant qu'elle n'a pas redéposé ses baguettes.
>
> Puisque leur quête du savoir leur apparaît primordiale, ces quatre philosophes veulent s'entendre sur une procédure à suivre afin qu'à tout moment au moins une personne du groupe réfléchisse. Or, absorbés par leurs lectures et n'ayant guère le temps de discuter de telles préoccupations primaires, les quatre philosophes ne savent jamais si leurs collègues ont faim ou non. Comment peuvent-ils procéder ?

[1] Nous donnons en fait une légère variante du problème original intitulé « The problem of the Dining Quintuple » [Dij87] par Dijkstra en 1965, et plus tard renommé *problème du dîner des philosophes* (« The Dining Philosophers Problem ») par l'informaticien Tony Hoare [Hoa85].

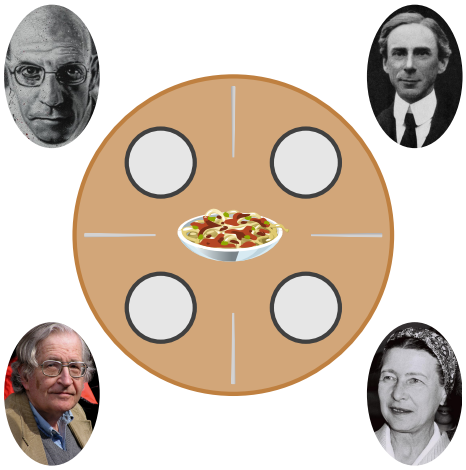


Figure 1.1 – Illustration du problème du dîner des philosophes.

Ces philosophes pourraient s'entendre pour toujours procéder de la façon suivante : lorsqu'une personne a faim, elle attend que la baguette à sa gauche devienne disponible, puis la prend ; elle attend que la baguette à sa droite devienne disponible, puis la prend, mange, et se remet au travail. Cette stratégie comporte toutefois un danger. Si les quatre personnes n'ont essentiellement jamais faim au même moment, leurs travaux de réflexion pourront se poursuivre sans problème. Par contre, imaginons que tous les philosophes aient faim en même temps. La première personne saisit la baguette à sa gauche, puis la deuxième, assez rapide, saisit également la baguette à sa gauche, puis pareillement pour la troisième et la quatrième personne. À ce moment, chacun des philosophes a sa baguette gauche en main et attend que sa baguette droite devienne disponible, ce qui ne se produira jamais. Leurs réfléxions ne se poursuivront jamais. Les philosophes se retrouvent dans une situation dite d'*interblocage*. La procédure, c'est-à-dire, l'*algorithme*, établie par le groupe donne lieu à une *situation de compétition*, aussi appelée *concurrence critique*. Autrement dit, le bon fonctionnement de la procédure dépend de l'ordre dans lequel les opérations sont exécutées par les *processus* ; les quatre philosophes.

Cet exemple ludique illustre les défis inhérents qui surgissent dans le développement de systèmes concurrents, autrement dit, de systèmes où plusieurs processus ou acteurs effectuent des opérations sur des *ressources partagées*, ici les baguettes.

Par exemple, que faire si une personne au rez-de-chaussée d'un immeuble tente de prendre l'ascenseur jusqu'au dernier étage, alors qu'une personne, au dernier étage, tente essentiellement au même moment, de descendre au premier étage ? Comment procéder lorsque plusieurs personnes tentent d'acheter les derniers billets d'un concert sur un site d'achat en ligne ? Est-il possible que plusieurs processus d'un logiciel tentent d'accéder à une même adresse mémoire et de modifier son contenu dans un ordre que sa développeuse n'avait pas cru possible, créant ainsi un *bogue* ?

Une mauvaise gestion d'éventuelles situations de compétition peut avoir des impacts considérables sur le bon fonctionnement d'un système ou d'un programme concurrent. Par exemple, un logiciel peut boucler à l'infini et ainsi se bloquer ; un programme peut entrer dans un état incohérent où des comportements inattendus se produisent, ou bien où des attaques informatiques deviennent possibles. Ces problèmes ont parfois d'énormes répercussions. En 2003, une panne de courant majeure toucha environs dix millions de personnes en Ontario au Canada et quarante-cinq millions de résidents du Midwest et du Nord-Est des États-Unis. Cette panne, l'une des plus grandes jamais connue, eut lieu suite à une surchage électrique qui mena à une situation de compétition dans un système de contrôle, ne déclenchant ainsi jamais d'alarme afin d'avertir les opérateurs du réseau qui auraient pu redistribuer le courant [Pou04]. De façon plus dramatique, la machine de radiothérapie *Therac-25*, produite par la société d'Énergie atomique du Canada Limited et en partie par la compagnie française CGR, fut impliquée, de 1985 à 1987, dans six incidents ayant causé la mort ou des blessures graves chez six patients atteints du cancer, suite à une trop grande exposition de radiations. Les dysfonctionnements de la machine Therac-25 furent attribués en grande partie à des problèmes de concurrence critique dans le logiciel de contrôle de l'appareil [Lev95]. Heureusement, les problèmes de développement de ce genre sont rarement de cette ampleur. Néanmoins, ils peuvent compromettre temps, fiabilité, sécurité informatique et ressources financières, par exemple. De plus, comme ces problèmes surgissent dans un système suite à une séquence d'exécution bien précise, il est souvent difficile pour un humain de les

cerner, même en testant minutieusement le système de façon empirique avant son déploiement.

Dans cette thèse, nous nous intéresserons aux méthodes formelles, c'est-à-dire mathématiques et algorithmiques, qui permettent de prouver le bon fonctionnement de systèmes concurrents, ou à défaut, de détecter des erreurs de façon automatisée. En particulier, nous nous intéresserons à la *complexité calculatoire* associée à ces méthodes et problèmes : comment vérifier rapidement le bon fonctionnement d'un système ; combien d'unités de temps et de mémoire sont théoriquement requises ou suffisantes afin d'accomplir ces tâches ?

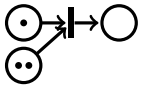
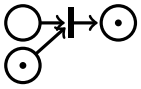
1.1 Solution et modélisation du problème du dîner des philosophes

Il existe plus d'une solution au problème du dîner des philosophes. Nous en décrivons une. Chaque baguette se voit attribuée un nombre entre 1 et 4. Par exemple, à la figure 1.1, les baguettes situées au bas, à droite, en haut et à gauche sont respectivement numérotées par 1, 2, 3 et 4. L'algorithme sur lequel les philosophes s'entendent est le suivant : lorsqu'une personne a faim, elle attend que la baguette adjacente numérotée par le plus petit nombre devienne disponible, puis la prend ; elle attend que l'autre baguette devienne disponible, puis la prend, mange, et se remet au travail. Cet algorithme est quasi identique à celui décrit plus tôt, à l'exception que le philosophe assis dans le coin inférieur gauche tente maintenant de prendre la baguette à sa droite en premier lieu. Grâce à cette légère modification, le scénario problématique décrit pour la solution présentée précédemment ne peut plus se produire. En effet, il est impossible que les quatre philosophes aient chacun une seule baguette en main. Autrement, un philosophe tiendrait uniquement la baguette 4, alors qu'il aurait dû attendre que la baguette 1 ou 3 soit disponible, selon sa disposition à table.

Néanmoins, afin d'être convaincu que cette solution est correcte, il ne suffit pas de prouver qu'un scénario problématique particulier ne peut pas se produire, mais bien qu'*aucun* tel scénario n'est possible. À cette fin, nous nous situons dans

le cadre de la *vérification formelle* et nous adoptons l'une de ses approches les plus répandues, la *vérification de modèles*, mieux connue sous son nom anglais : le « model checking ». Cette approche consiste à modéliser un système concret par un *modèle* mathématique. Les propriétés qui doivent être vérifiées sont exprimées sous forme logique, c'est ce qu'on appelle la *spécification* du système. Afin de vérifier que le système se comporte correctement, on met au point un algorithme qui vérifie que le modèle respecte sa spécification.

Les *réseaux de Petri* constituent l'un des modèles mathématiques les plus répandus permettant de modéliser les systèmes concurrents. Depuis leur introduction par Carl Adam Petri en 1962 [Pet62], les réseaux de Petri ont été étudiés dans des milliers d'articles scientifiques, et ont trouvé des applications non seulement dans la modélisation de programmes concurrents, mais également dans la modélisation de systèmes opérationnels, biologiques et chimiques [HGD08, RLM96, van98, GS92, BCR01]. En particulier, les réseaux de Petri permettent de modéliser notre solution au problème du dîner des philosophes. Nous illustrons une telle modélisation à la figure 1.2.

Un réseau de Petri est identifié par ses *places*, représentées par des cercles ◯, ses *transitions*, représentées par des rectangles ▮, et ses *arcs*, représentés par des flèches ⟶, qui relient les places et les transitions. Chacune des places d'un réseau de Petri peut être *marquée* par un ou plusieurs jetons, représentés par des petits cercles •. Une transition peut être *activée* si *chaque* place, qui lui est reliée par un arc entrant, est marquée par au moins un jeton. Par exemple, la transition du réseau de Petri ⊙⟶▮⟶◯ peut être activée, puisque son unique place d'entrée, située à gauche, contient un jeton. Lorsque l'on active une transition, chaque place d'entrée perd un jeton, et chaque place de sortie gagne un jeton. Par exemple, en activant la transition décrite ci-haut, la place de gauche perd son jeton et la place de droite en gagne un : ◯⟶▮⟶⊙, il n'est ici plus possible d'activer cette transition à nouveau. À titre d'exemple supplémentaire, la transition du réseau de Petri  peut être activée et mène à  .

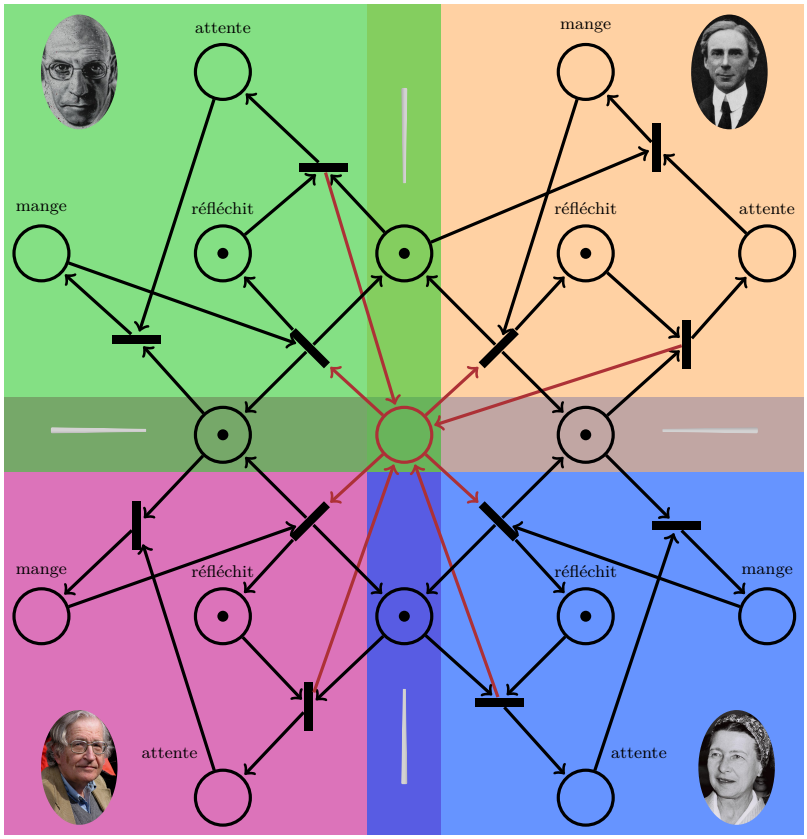


Figure 1.2 – Solution au problème du dîner des philosophes modélisée par un réseau de Petri.

La concurrence d'un système apparaît ainsi dans un réseau de Petri lorsqu'une place « entre » dans plusieurs transitions, puisque l'activation de l'une de ces transitions peut empêcher l'activation d'une autre. Intuitivement, les places représentent donc des ressources ou des états locaux, les jetons représentent une quantité de ressources disponibles, et les transitions représentent des actions.

Nous sommes maintenant en mesure d'expliquer notre modélisation de la solution au problème du dîner des philosophes illustrée à la figure 1.2. Chaque place située devant chaque baguette indique par un jeton si la baguette est disponible ou non. Chaque philosophe est représenté par trois états locaux indiquant respectivement s'il *réfléchit*, s'il tient une baguette et est donc en *attente*, ou s'il *mange*. Une place additionnelle située au centre compte le nombre de philosophes qui ne

réfléchissent actuellement pas. Considérons le philosophe situé dans le coin inférieur gauche ( ). Initialement, celui-ci réfléchit. Tel que décrit plus tôt, ce philosophe priorise la baguette située à sa droite. Ainsi, s'il a faim et que la baguette située à sa droite est disponible, il peut activer la transition située à côté de la baguette. Si cette transition est activée, le philosophe est maintenant en attente, et ainsi un jeton est ajouté à la place du centre de la table. Lorsqu'il est en attente et que sa baguette gauche est disponible, le philosophe peut activer la transition située à côté de cette baguette, ce qui amène un jeton dans la place qui indique que le philosophe mange. Quand le philosophe n'a plus faim, il peut activer la transition devant lui, ce qui rend ses deux baguettes à nouveaux disponibles, et retire un jeton de la place du centre, indiquant qu'il y a un philosophe de moins qui n'est pas au travail.

Le problème se formalise maintenant de la façon suivante : à partir de la configuration initiale représentée à la figure 1.2, est-ce que toute séquence d'activation de transitions maintient trois jetons ou moins dans la place du centre ? Autrement dit, l'algorithme comporte une erreur si, et seulement si, il existe une séquence d'activation qui mème à quatre jetons (ou plus) dans la place du centre. Cela est une instance du *problème de couverture*, et de façon plus générale, du *problème d'accessibilité*.

Dans cette thèse, nous étudierons le problème de couverture, le problème d'accessibilité ainsi que certaines de leurs variantes. Il est connu que ces problèmes sont théoriquement solubles par un ordinateur. Cependant, ils ne sont pas simples à résoudre en pratique, et d'un point de vue théorique, la complexité calculatoire précise du problème d'accessibilité est, à ce jour, encore inconnue depuis plus de quarante ans. Dans cette thèse, nous établirons la complexité précise du problème d'accessibilité, sous certaines contraintes, dans le formalisme équivalent des *systèmes d'addition de vecteurs*. Cela nous permettra ainsi de repousser la frontière des connaissances calculatoires actuelles sur ce problème. Nous introduirons également une approche théorique qui permet de résoudre le problème de couverture *rapidement* en pratique. Cette approche sera implémentée concrètement dans un pro-

gramme informatique nommé QCover, qui surpasse, à bien des égards, d'autres programmes. QCover permet bien sûr de démontrer que la solution au problème du dîner des philosophes est correcte, mais au-delà de cet exemple pédagogique, QCover permet de vérifier des réseaux de Petri possédant des milliers de places et de transitions, et modélisant des systèmes concrets tels que des programmes concurrents, des systèmes de production ou des protocoles de communication. Cette étude sera complétée par une investigation plus théorique de différents problèmes dans les *systèmes de transitions bien structurés*, une abstraction mathématique des réseaux de Petri et des systèmes d'addition de vecteurs.

1.2 Contributions détaillées et organisation de la thèse

Cette thèse présente trois contributions principales, publiées en majeure partie dans [BFM14, BFG+15, BFHH16, BFM16], et décrites dans trois chapitres principaux :

1) Au chapitre 3, nous étudions les systèmes de transitions bien structurés. Ce modèle, introduit par Finkel [Fin87a, Fin87b], puis étudié par Abdulla *et al.* [ACJT96, ACJT00], permet d'abstraire non seulement les réseaux de Petri et leurs extensions, mais également une grande variété de modèles de sytèmes infinis dont des modèles de systèmes à canaux, d'algèbres de processus, de systèmes de réécriture et d'automates temporisés. Les systèmes de transitions bien structurés permettent d'étudier l'algorithmique d'un large éventail de modèles, qui surgissent en vérification formelle, de façon générique sans se soucier de leurs détails propres. Par exemple, la possibilité de résoudre algorithmiquement le problème de couverture pour les réseaux de Petri peut être démontrée de façon beaucoup plus générale dans le contexte des systèmes de transitions bien structurés. Nous considérons les systèmes de transitions bien structurés, dits à *branchement infini*, c'est-à-dire, les systèmes où un état x peut posséder une infinité d'états successeurs. Par exemple, ces systèmes incluent une extension des réseaux de Petri où certaines transitions peuvent consommer ou produire

un nombre arbitraire de jetons. Les systèmes de transitions bien structurés à branchement infini n'ont jamais été étudiés dans leur cadre général, nous développons ainsi des outils mathématiques afin de les analyser. Grâce à ces outils, nous étudions leurs principaux problèmes de vérification considérés dans la littérature, dont le problème de couverture, et nous délimitons les hypothèses suffisantes et nécessaires afin de rendre possible ou impossible leur résolution à l'aide d'un ordinateur (c.-à-d. décidabilité c. indécidabilité).

2) Au chapitre 4, nous étudions les systèmes d'addition de vecteurs, un modèle de machines à compteurs équivalent, en terme d'expressivité et de calculabilité, aux réseaux de Petri. Les systèmes d'addition de vecteurs sont parfois préférés aux réseaux de Petri dans la modélisation, dû à leur structure d'automates, ou dans les preuves mathématiques, dû à la possibilité de les représenter par des ensemble finis de vecteurs. Nous étudions le problème d'accessibilité qui consiste à déterminer si une configuration y est accessible à partir d'une configuration initiale x. Ce problème central en vérification formelle est connu comme soluble par un ordinateur. Cependant, la complexité calculatoire précise du problème n'est pas connue, à l'exception du cas à un seul compteur qui est NP-complet [HKOW09]. Nous établissons ici la complexité précise du problème dans le cas des systèmes à *deux* compteurs, plus précisément, sa PSPACE-complétude, résolvant ainsi un problème ouvert depuis 1986 [HRHY86].

3) Au chapitre 5, nous étudions le problème de couverture dans les réseaux de Petri qui consiste à déterminer, étant donné deux marquages x et y des places d'un réseau de Petri, s'il existe une séquence d'activation des transitions qui mène de x vers un certain marquage y' où chaque place contient au moins autant de jetons dans y' que y. Nous revisitons l'*algorithme arrière*, bien connu dans le domaine des systèmes de transitions bien structurés, et nous l'adaptons en exploitant des avancées récentes sur les réseaux de Petri dits *continus* [FH15]. Nous montrons que cette approche est compétitive en pratique en l'implémentant dans l'outil QCover.

Cette thèse comprend six chapitres. Au chapitre 2, nous introduisons les notions préliminaires nécessaires à la lecture de la thèse. Aux chapitres 3, 4 et 5 nous présentons les contributions mentionnées ci-haut. Chacun de ces trois chapitres se termine par une discussion. Nous concluons, au chapitre 6, par un retour global sur cette thèse et des suggestions de travaux futurs.

2

PRÉLIMINAIRES

Dans ce chapitre, nous introduisons l'ensemble des concepts, définitions et notations d'ordre général ou commune à plusieurs chapitres. Les sections 2.4, 2.8 et 2.9 portent respectivement sur les ordres, les systèmes de transitions et les machines à compteurs, et sont particulièrement importantes à la compréhension des chapitres subséquents.

2.1 Ensembles

Nous dénotons $\mathbb{N} = \{0, 1, 2, \ldots\}$ l'ensemble des entiers naturels, $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ l'ensemble des entiers et $\mathbb{Q}$ l'ensemble des nombres rationnels. Pour tout ensemble $\mathbb{A} \subseteq \mathbb{Q}$, nous définissons $\mathbb{A}_{\geq 0} \stackrel{\text{déf}}{=} \{a \in \mathbb{A} : a \geq 0\}$ et $\mathbb{A}_{>0} \stackrel{\text{déf}}{=} \{a \in \mathbb{A} : a > 0\}$. Pour tous $a, b \in \mathbb{N}_{>0}$, nous écrivons $[a, b] \stackrel{\text{déf}}{=} \{a, a+1, \ldots, b\}$, $[a] \stackrel{\text{déf}}{=} [1, a]$ et $[a, \infty) \stackrel{\text{déf}}{=} \{a, a+1, \ldots\}$.

2.2 Matrices et vecteurs

Nous prenons le soin de définir les matrices et les vecteurs, ainsi que certaines notations qui nous seront utiles.

Soient $\mathbb{A}, \mathbb{D} \subseteq \mathbb{Q}$ des ensembles tels que $\mathbb{D}$ est clos sous l'addition, c.-à-d. $\mathbb{D}+\mathbb{D} \stackrel{\text{déf}}{=} \{x + y : x, y \in \mathbb{D}\} \subseteq \mathbb{D}$, et clos sous la multiplication par un scalaire, c.-à-d. $\mathbb{A}\cdot\mathbb{D} \stackrel{\text{déf}}{=} \{a \cdot x : a \in \mathbb{A}, x \in \mathbb{D}\} \subseteq \mathbb{D}$. Soient E et F des ensembles finis. Nous dénotons $\mathbb{D}^{E\times F}$ l'ensemble des *matrices* dont les lignes et les colonnes sont respectivement indicées par E et F. Autrement dit, $\mathbb{D}^{E\times F}$ est l'ensemble des fonctions de $E \times F$ vers $\mathbb{D}$, muni de l'addition « $+$ » et de la multiplication par un scalaire « $\cdot$ » définies, pour tous $\mathbf{A}, \mathbf{B} \in \mathbb{D}^{E\times F}$ et pour tout $a \in \mathbb{A}$, par

- $\mathbf{A} + \mathbf{B} \stackrel{\text{déf}}{=} \mathbf{C}$ tel que $\mathbf{C}(e, f) = \mathbf{A}(e, f) + \mathbf{B}(e, f)$ pour tous $e, f \in E$,
- $a \cdot \mathbf{A} \stackrel{\text{déf}}{=} \mathbf{C}$ tel que $\mathbf{C}(e, f) = a \cdot \mathbf{A}(e, f)$ pour tous $e, f \in E$.

Soient $d, d' \in \mathbb{N}$, par abus de notation, nous écrivons simplement $\mathbb{D}^{d \times d'}$ pour $\mathbb{D}^{[d] \times [d']}$. Lorsque $E = \{e_1, e_2, \ldots, e_d\}$ et $F = \{f_1, f_2, \ldots, f_{d'}\}$ sont, respectivement, totalement ordonnés par $e_1 < e_2 < \ldots < e_d$ et $f_1 < f_2 < \ldots < f_d$, nous représentons $\mathbf{A} \in \mathbb{D}^{E \times F}$ par

$$\begin{pmatrix} \mathbf{A}(e_1, f_1) & \mathbf{A}(e_1, f_2) & \ldots & \mathbf{A}(e_1, f_{d'}) \\ \mathbf{A}(e_2, f_1) & \mathbf{A}(e_2, f_2) & \ldots & \mathbf{A}(e_2, f_{d'}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}(e_d, f_1) & \mathbf{A}(e_d, f_2) & \ldots & \mathbf{A}(e_d, f_{d'}) \end{pmatrix}.$$

En particulier, nous représentons $\mathbf{A} \in \mathbb{D}^{d \times d'}$ par

$$\begin{pmatrix} \mathbf{A}(1, 1) & \mathbf{A}(1, 2) & \ldots & \mathbf{A}(1, d') \\ \mathbf{A}(2, 1) & \mathbf{A}(2, 2) & \ldots & \mathbf{A}(2, d') \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}(d, 1) & \mathbf{A}(d, 2) & \ldots & \mathbf{A}(d, d') \end{pmatrix}.$$

Soient $E' \subseteq E$ et $F' \subseteq F$, nous écrivons $\mathbf{A}[E' \times F']$ afin de représenter la sous-matrice $\mathbf{A}$ restreinte à $E' \times F'$, c.-à-d. la matrice $\mathbf{A}' \in \mathbb{D}^{E' \times F'}$ telle que $\mathbf{A}'(e, f) = \mathbf{A}(e, f)$ pour tout $e \in E'$ et tout $f \in F'$. Nous étendons naturellement l'addition de matrices aux ensembles $A, B \subseteq \mathbb{D}^{E \times F}$ en définissant $A + B \stackrel{\text{déf}}{=} \{\mathbf{A} + \mathbf{B} : \mathbf{A} \in A, \mathbf{B} \in B\}$.

Un *vecteur* est une matrice telle que $|F| = 1$, c.-à-d. avec une seule colonne. Afin d'alléger la notation, nous omettons F lorsqu'il ne contient qu'un élément. Ainsi, l'ensemble des vecteurs indicés par E est dénoté $\mathbb{D}^E$, et le vecteur $\boldsymbol{v} \in \mathbb{D}^E$ restreint à $E' \subseteq E$ est dénoté $\boldsymbol{v}[E']$. Nous dénotons le *support* de $\boldsymbol{v} \in \mathbb{D}^E$ par $[\![\boldsymbol{v}]\!] = \{e \in E : \boldsymbol{v}(e) > 0\}$. Similairement au cas des matrices, lorsque $E = \{e_1, e_2, \ldots, e_d\}$

est totalement ordonné par $e_1 < e_2 < \ldots < e_d$, nous représentons $\boldsymbol{v} \in \mathbb{D}^E$ par

$$\begin{pmatrix} \boldsymbol{v}(e_1) \\ \boldsymbol{v}(e_2) \\ \vdots \\ \boldsymbol{v}(e_d) \end{pmatrix}$$

ou $(\boldsymbol{v}(e_1), \boldsymbol{v}(e_2), \ldots, \boldsymbol{v}(e_d))$[1]. Nous définissons la multiplication entre une matrice $\mathbf{A} \in \mathbb{D}^{E \times F}$ et un vecteur $\boldsymbol{u} \in \mathbb{D}^F$ comme étant le vecteur $\boldsymbol{v} \in \mathbb{D}^E$ tel que, pour tout $e \in E$,

$$\boldsymbol{v}(e) = \sum_{f \in F} \mathbf{A}(e, f) \cdot \boldsymbol{u}(f) \ .$$

La *norme* d'une matrice $\mathbf{A} \in \mathbb{D}^{E \times F}$ est définie par

$$\|\mathbf{A}\| \stackrel{\text{déf}}{=} |F| \cdot \max\{|\mathbf{A}(e, f)| : e, f \in E\} \ .$$

En particulier, la norme d'un vecteur $\boldsymbol{v} \in \mathbb{D}^E$ est définie par $\|\boldsymbol{v}\| \stackrel{\text{déf}}{=} \max\{|\boldsymbol{v}(e)| : e \in E\}$. Nous étendons naturellement la norme à un ensemble fini $V \subseteq \mathbb{D}^E$ par $\|V\| \stackrel{\text{déf}}{=} \max\{\|\boldsymbol{v}\| : \boldsymbol{v} \in V\}$.

2.3 Langages formels

Soit Σ un ensemble dénombrable, dans certains contextes, nous dirons que Σ est un *alphabet*. Les éléments de Σ sont apppelés des *lettres*. Un *mot fini* w, ou simplement *mot*, est une suite finie de lettres $\sigma_1, \sigma_2, \ldots, \sigma_n$ dénotée par la concaténation $w = \sigma_1\sigma_2 \cdots \sigma_n$. Un *mot infini* est une suite infinie de lettres $\sigma_1, \sigma_2, \ldots$ dénotée par la concaténation $w = \sigma_1\sigma_2 \cdots$ où cette notation représente la fonction $u_w : \mathbb{N}_{>0} \to \Sigma$ telle que $u_w(i) = \sigma_i$. Nous dénotons ε le mot vide, c'est-à-dire, la suite vide. Nous définissons Σ^* et Σ^ω comme étant, respectivement, l'ensemble des mots finis et l'ensemble des mots infinis. Nous posons $\Sigma^\infty \stackrel{\text{déf}}{=} \Sigma^* \cup \Sigma^\omega$. Nous disons

[1] Cela ne causera pas d'ambiguïté puisque les vecteurs lignes ne seront jamais utilisés dans cette thèse.

que $L \subseteq \Sigma^*$ est un *langage*. Soient $w \in \Sigma^*$ et $\sigma \in \Sigma$, nous dénotons par $|w|$ la longueur de w et par $|w|_\sigma$ le nombre d'occurrences de σ dans w. Soient $w \in \Sigma^*$ et $1 \leq i \leq |w|$, w_i dénote la $i^{\text{ème}}$ lettre de w. Soient $w \in \Sigma^\omega$ et $i \in \mathbb{N}_{>0}$, w_i dénote la $i^{\text{ème}}$ lettre de w.

Soit Σ un alphabet fini. L'*image de Parikh* d'un mot $w \in \Sigma^*$, dénotée par $\mathsf{Parikh}(w) \in \mathbb{N}^\Sigma$, est le vecteur tel que $\mathsf{Parikh}(w)(\sigma) = |w|_\sigma$ pour tout $\sigma \in \Sigma$. L'image de Parikh d'un mot w est donc son image commutative, c'est-à-dire, une représentation des lettres qui apparaissent dans w sans égard à leur ordre.

2.4 Relations et ordres

Soit un ensemble X. Nous dénotons Id la relation identité sur X, c.-à-d. $\mathsf{Id} \stackrel{\text{déf}}{=} \{(x, x) : x \in X\}$. Nous disons qu'une relation $\leq \subseteq X \times X$ est un *préordre* si $\leq$ est réflexive et transitive, c.-à-d. pour tous $x, y, z \in X$, $x \leq x$ et $x \leq y \wedge y \leq z \implies x \leq z$. Si $\leq$ est de surcroît antisymétrique, c.-à-d. $x \leq y \wedge y \leq x \implies x = y$, nous disons que $\leq$ est un *ordre partiel*.

2.4.1 Beaux préordres

Un préordre (resp. ordre partiel) $\leq$ est un *beau préordre* (resp. *bel ordre*) si pour toute suite infinie $x_0, x_1, \ldots$ d'éléments de X, il existe $i, j \in \mathbb{N}$ tels que $i < j$ et $x_i \leq x_j$. Les ensembles munis d'un beau préordre jouissent d'un certain nombre de propriétés. Nous détaillons quelques-unes de ces propriétés dans cette sous-section et la suivante. Pour une présentation plus détaillée, voir par exemple [SS12].

En particulier, tout ensemble muni d'un beau préordre ne possède pas d'*anti-chaîne* infinie, c'est-à-dire, de sous-ensemble infini d'éléments deux à deux incomparables. Il existe plusieurs façons de construire des ensembles munis de beaux préordres. Par exemple, l'égalité est un beau préordre pour un ensemble Q si, et seulement si, Q est fini. L'ordre usuel $\leq$ sur $\mathbb{N}$ est également un beau préordre. Soient X et Y des ensembles respectivement munis des beaux préordres $\leq_X$ et $\leq_Y$.

Il est bien connu que $\leq_{X\times Y}$ est un beau préordre pour $X \times Y$, où

$$(x, y) \leq_{X\times Y} (x', y') \iff (x \leq_X x') \wedge (y \leq_Y y') \ .$$

Ainsi, l'ordre $\leq_{\mathbb{N}^d}$ défini sur $\mathbb{N}^d$ par

$$(x_1, x_2, \ldots, x_d) \leq_{\mathbb{N}^d} (y_1, y_2, \ldots, y_d) \iff \forall i \in [d], \ \ x_i \leq y_i$$

est un beau préordre, et du coup, un bel ordre. Ce résultat est connu dans la littérature sous le nom du lemme de Dickson [Dic13]. Notons que l'ordre usuel $\leq_{\mathbb{Z}}$ sur $\mathbb{Z}$ n'est pas un beau préordre, en effet la suite $0, -1, -2, \ldots$ est infiniment décroissante. Nous étendons $\mathbb{N}$ à $\mathbb{N}_\omega \stackrel{\text{déf}}{=} \mathbb{N} \cup \{\omega\}$ et nous étendons $\leq_{\mathbb{N}}$ à $\leq_{\mathbb{N}_\omega} \stackrel{\text{déf}}{=} \leq_{\mathbb{N}} \cup \{(x, \omega) : x \in \mathbb{N}_\omega\}$, autrement dit ω borne supérieurement tous les éléments de $\mathbb{N}_\omega$. Similairement à $\mathbb{N}$, l'ensemble $\mathbb{N}_\omega$ s'étend naturellement à $\mathbb{N}_\omega^d$, et $\leq_{\mathbb{N}_\omega^d}$ est également un beau préordre. Nous écrirons simplement $\leq$ lorsque le domaine n'est pas ambigu. Afin d'alléger la notation, nous écrivons naturellement

$$\begin{aligned}
x \not\leq y \ &\text{ lorsque } \neg(x \leq y) \ , \\
x \geq y \ &\text{ lorsque } \quad y \leq x \ , \\
x \not\geq y \ &\text{ lorsque } \quad y \not\leq x \ , \\
x < y \ &\text{ lorsque } \quad x \leq y \wedge y \not\leq x \ , \\
x \not< y \ &\text{ lorsque } \neg(x < y) \ , \\
x > y \ &\text{ lorsque } \quad y < x \text{ , et} \\
x \not> y \ &\text{ lorsque } \quad y \not< x \ .
\end{aligned}$$

Par exemple, sur $\mathbb{N}^3$,

$$(2,1,2) \leq (2,3,2)\ ,$$
$$(2,1,2) < (2,3,2)\ ,$$
$$(2,1,2) > (0,1,1)\ \text{, et}$$
$$(2,1,2) \text{ est incomparable avec } (0,3,2)\ .$$

Au chapitre 3, nous considérerons également l'*ordre sous-mot* $\preceq$ sur les mots finis. Soient Σ un alphabet fini et $u, v \in \Sigma^*$ des mots, nous avons $u \preceq v$ si, et seulement si, u peut être obtenu à partir de v en lui retirant zéro, une ou plusieurs lettres. Par exemple, si $\Sigma = \{a, b, c\}$, alors $ab \preceq ac\boldsymbol{ab}$, $ab \preceq b\boldsymbol{a}ac\boldsymbol{b}$, $abc \preceq \boldsymbol{abc}$, et $\varepsilon \preceq aaa$, mais $ca \not\preceq baacb$. L'ordre sous-mot est un beau préordre ; ce résultat est connu dans la littérature sous le nom du lemme de Higman [Hig52].

Notons que lorsque nous travaillerons sur un produit cartésien $Q \times X$ où Q est un ensemble fini ordonné par l'égalité, et X est un ensemble muni d'un beau préordre, nous dénoterons les éléments $(q, x) \in Q \times X$ par $q(x)$ afin d'alléger la notation.

2.4.2 Ensembles clos par le haut/bas et bases

Soient X un ensemble muni d'un préordre $\leq$, et $A \subseteq X$. Nous définissons la *clôture par le bas* de A par $\downarrow A \stackrel{\text{déf}}{=} \{x \in X : \exists y \in A \text{ t.q. } x \leq y\}$. De façon similaire, nous définissons la *clôture par le haut* de A par $\uparrow A \stackrel{\text{déf}}{=} \{x \in X : \exists y \in A \text{ t.q. } x \geq y\}$. Nous disons que $A \subseteq X$ est *clos par le bas* si $A = \downarrow A$, et que A est *clos par le haut* si $A = \uparrow A$. Soit $x \in X$, nous écrivons simplement $\downarrow x$ pour $\downarrow\{x\}$, et $\uparrow x$ pour $\uparrow\{x\}$. Soit $A_1, A_2, \ldots \subseteq X$ une suite infinie de sous-ensembles de X, il est possible de montrer que les chaînes d'inclusion

$$\uparrow A_1 \subseteq \uparrow A_2 \subseteq \ldots \quad \text{et} \quad \downarrow A_1 \supseteq \downarrow A_2 \supseteq \ldots$$

se stabilisent en un nombre fini d'étapes si, et seulement si, $\leq$ est un beau préordre. Autrement dit, dans chacun des cas, il existe $n \in \mathbb{N}_{>0}$ tel que pour tout $i \geq n$, $A_i = A_n$. Nous utiliserons ce fait à quelques reprises dans les chapitres subséquents.

Une *base* d'un ensemble clos par le haut $A \subseteq X$ est un ensemble $B \subseteq A$ tel que $A = \uparrow B$. Il est connu qu'un ensemble clos par le haut possède toujours une base finie minimale pour l'inclusion lorsque $\leq$ est un beau préordre. De plus, lorsque $\leq$ est un bel ordre, il existe une *unique* base minimale. Par exemple, soit $A \subseteq \mathbb{N}^2$ tel que $A = \{(a, b) : a + b \geq 2\}$, alors A est clos par le haut et

$$A = \uparrow \{(0, 2), (1, 1), (2, 0)\}.$$

Soit $A \subseteq X$. Lorsque $\leq$ est un bel ordre, nous dénotons l'unique *base minimale* de $\uparrow A$ par

$$\mathsf{MinBase}(A) \stackrel{\text{déf}}{=} \text{unique base minimale de } \uparrow A \ .$$

Cette notation a l'avantage de donner la base minimale de A lorsque A est clos par le haut et de minimiser A lorsque A est implicitement considéré comme la base d'un ensemble clos par le haut. Par exemple, $\mathsf{MinBase}(\mathbb{N}^2) = \{(0, 0)\}$ et $\mathsf{MinBase}(\{(1, 1), (1, 2), (2, 0))\}) = \{(1, 1), (2, 0)\}$. Afin de minimiser une base finie B, il suffit de comparer les éléments de B deux à deux et d'éliminer les éléments de B pour lesquels il existe un autre élément de B plus petit. Autrement dit, lorsque B est fini, nous avons $\mathsf{MinBase}(B) = \{x \in B : \forall y \in B \setminus \{x\}, y \not\leq x\}$. Notons que nous exploitons le fait que $\leq$ est un bel ordre, puisque, sans antisymétrie, plusieurs éléments de B peuvent être équivalents, c.-à-d., il peut exister des éléments $x, y \in B$ tels que $x \neq y$ et $x \leq y \leq x$.

Le concept de base peut être défini similairement pour les ensembles clos par le bas, mais l'existence d'une base finie n'est plus garantie lorsque $\leq$ est un beau préordre ; p. ex. considérez $\mathbb{N}$ qui est lui-même clos par le bas. Au chapitre 3, nous verrons qu'il existe des conditions suffisantes à l'obtention de description finie de tout ensemble clos par le bas.

2.4.3 Fonctions monotones

Soit X un ensemble ordonné par $\leq$. Une *fonction partielle* $f : X \to X$ est une fonction dont le domaine est un sous-ensemble de X, c.-à-d. $f : X \to X$ n'est pas nécessairement définie pour tout $x \in X$. Lorsque f est définie pour tout $x \in X$, nous disons simplement que f est une *fonction*. Une fonction partielle $f : X \to X$ est dite *non décroissante* (resp. *croissante*) si, pour tout $x \in X$, lorsque $f(x)$ est définie

- $f(y)$ est définie pour tout $y \geq x$, et
- $x < y \implies f(x) \leq f(y)$ (resp. $x < y \implies f(x) < f(y)$).

2.5 Cônes et ensembles semi-linéaires

Soient $\boldsymbol{b} \in \mathbb{Q}^d$ et $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\} \subseteq \mathbb{Q}^d$ et $\mathbb{D} \subseteq \mathbb{Q}$. Le $\mathbb{D}$-*cône engendré par* P est défini par

$$\mathsf{Cone}_{\mathbb{D}}(P) \stackrel{\text{déf}}{=} \left\{ \sum_{i \in [n]} \lambda_i \cdot \boldsymbol{p}_i : \lambda_i \in \mathbb{D} \right\} .$$

Nous dénotons $\mathsf{Lin}_{\mathbb{D}}(\boldsymbol{b}, P) \stackrel{\text{déf}}{=} \boldsymbol{b} + \mathsf{Cone}_{\mathbb{D}}(P)$. Nous ne spécifions pas l'indice $\mathbb{D}$ lorsque $\mathbb{D} = \mathbb{N}$.

Un ensemble $A \subseteq \mathbb{Z}^d$ est dit *linéaire* s'il existe un $\boldsymbol{b} \in \mathbb{Z}^d$ et un ensemble fini $P \subseteq \mathbb{Z}^d$ tels que $A = \mathsf{Lin}(\boldsymbol{b}, P)$. Nous appelons $\boldsymbol{b}$ la *base* et les éléments de P les *périodes* de A. Nous dénotons $\mathsf{Lin}(\boldsymbol{b}, P)$ l'ensemble linéaire dont la base est $\boldsymbol{b}$ et l'ensemble de périodes est P. Un ensemble *semi-linéaire* est une union finie d'ensembles linéaires.

2.6 Théories logiques du premier ordre

Nous ferons brièvement appel à deux théories logiques du premier ordre dans cette thèse. Nous les introduisons de façon plutôt informelle. Le lecteur ou la lec-

trice peut se référer à [End01] pour une introduction plus détaillée et formelle des différentes théories logiques.

Soit $\mathbb{A} \in \{\mathbb{N}, \mathbb{Q}_{\geq 0}\}$. Nous dénotons par $\mathsf{FO}(\mathbb{A}, +, <)$ la théorie des formules logiques quantifiées sur $\mathbb{A}$ par des variables du premier ordre, et construites à partir des opérateurs booléens $\wedge, \vee, \neg$, de l'addition $+$, du prédicat $<$, et de constantes dans $\mathbb{A}$. Puisque $x = y$ est équivalent à $\neg(x < y) \wedge \neg(y < x)$, les prédicats $=$ et $\neq$ peuvent aussi être utilisés. Lorsqu'une formule ϕ possède d variables non quantifiées $x_1, x_2, \ldots, x_d$, le sous-ensemble de $\mathbb{A}^d$ décrit par ϕ est l'ensemble des valeurs qui satisfont ϕ, c.-à-d., $\{(x_1, \ldots, x_d) \in \mathbb{A}^d : (x_1, \ldots, x_d) \models \phi\}$.

Il est connu que la logique $\mathsf{FO}(\mathbb{N}, +, <)$, nommée *arithmétique de Presburger*, caractérise précisément les ensembles semi-linéaires [GS66]. De plus, l'arithmétique de Presburger [Pre29, Coo72] ainsi que $\mathsf{FO}(\mathbb{Q}_{\geq 0}, +, <)$ [FR75] sont décidables, c'est-à-dire, qu'il existe un algorithme qui sur entrée une formule ϕ de ces théories permet de décider si ϕ est satisfaisable.

2.7 Calculabilité et complexité du calcul

Dans cette section, nous survolons les notions de décidabilité et de calculabilité. Nous utilisons le modèle standard des machines de Turing. Des notions formelles de temps et de mémoire permettent ensuite de classifier les problèmes calculatoires selon leur complexité, c'est-à-dire la quantité de ressources nécessaires et suffisantes à leur résolution. Cette section ne se veut qu'un rappel. Pour un traitement beaucoup plus détaillé de la théorie de la calculabilité et de la complexité du calcul, il est possible de consulter [Sip06, Per14] sur lesquels cette section est basée.

2.7.1 Machines de Turing

Definition 2.1. Une *machine de Turing* est un septuplet $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_{\text{init}}, q_{\text{acc}}, q_{\text{rejet}})$ où

- Q est un ensemble fini (*états de contrôle*),
- Σ est un alphabet fini tel que $\sqcup \notin \Sigma$ (*alphabet d'entrée*),

- Γ est un alphabet fini tel que $\sqcup \in \Gamma$ et $\Sigma \subseteq \Gamma$ (*alphabet de travail*),
- $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{\text{gauche,droite}\}}$ (*fonction de transition*),
- $q_{\text{init}} \in Q$ (*état de contrôle initial*),
- $q_{\text{acc}} \in Q$ (*état de contrôle acceptant*), et
- $q_{\text{rej}} \in Q$ tel que $q_{\text{rej}} \neq q_{\text{acc}}$ (*état de contrôle rejetant*).

Nous disons que $\mathcal{M}$ est *déterministe* si $|\delta(q, \sigma)| \leq 1$ pour tout $(q, \sigma) \in Q \times \Gamma$. Autrement, $\mathcal{M}$ est dite *non déterministe*.

Une machine de Turing ressemble à un automate fini qui agit à titre d'unité de contrôle d'un *ruban* infini de mémoire pouvant être lu ou modifié par le biais d'une tête de lecture. À la figure 2.1, nous illustrons une machine de Turing dont l'unité de contrôle se situe au dessus ( ), dont le ruban se situe au bas ( ) et dont la tête de lecture apparaît sous la forme d'une flèche pointant sur la case mémoire actuellement considérée.

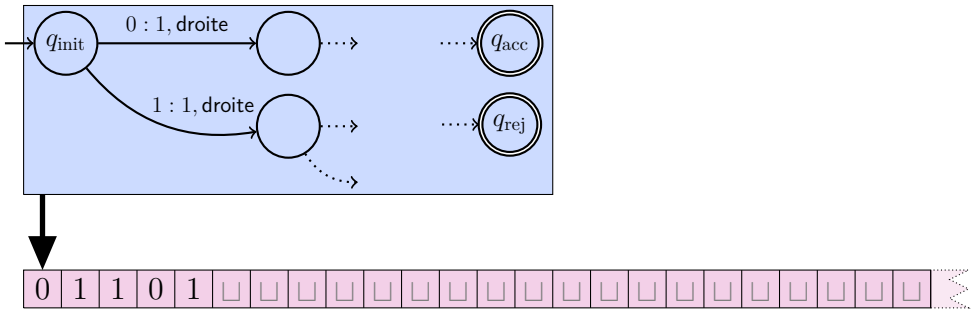


Figure 2.1 – Illustration d'une machine de Turing

Intuitivement, une machine de Turing débute dans l'état de contrôle initial et reçoit un mot $w \in \Sigma^*$ sur les $|w|$ premières cases de son ruban de mémoire. Toutes les autres cases contiennent le symbole vide $\sqcup$. Par exemple, à la figure 2.1, la machine débute dans l'état situé dans le coin supérieur gauche avec le mot $w = 01101$. Une *configuration* de la machine est un triplet $(p, u, i) \in Q \times \Gamma^\omega \times \mathbb{N}_{>0}$ où p est l'état de contrôle actuel, u est le mot infini contenu sur le ruban et i est

la position de la tête de lecture. À partir de la configuration (p, u, i), la machine peut emprunter une transition spécifiée par $\delta(p, u_i)$, s'il en existe une. Une telle transition $(q, \sigma, \sim)$ indique que la machine doit remplacer la case i du ruban par σ, se déplacer dans l'état de contrôle q, et déplacer sa tête de lecture d'une case dans la direction $\sim \in \{\mathsf{gauche}, \mathsf{droite}\}$. Dans le cas particulier où la tête est en position 1 et tente de se déplacer à gauche, la tête demeure stationnaire.

Nous associons à une machine de Turing $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_{\text{init}}, q_{\text{acc}}, q_{\text{rejet}})$ le système de transition $(Q \times \Gamma^\omega \times \mathbb{N}_{>0}, \to)$ tel que $(p, u, i) \to (q, v, j)$ si $p \notin \{q_{\text{acc}}, q_{\text{rej}}\}$ et s'il existe une transition qui mène de la configuration (p, u, i) à la configuration (q, v, j).

Les machines de Turing peuvent naturellement être étendues à $k > 1$ rubans [Sip06, Per14] en adaptant la définition de fonction de transition ; dans ce cas, l'ensemble des configurations devient $Q \times (\Gamma^\omega \times \mathbb{N}_{>0})^k$.

2.7.2 Calculabilité

Une machine de Turing $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_{\text{init}}, q_{\text{acc}}, q_{\text{rejet}})$ s'*arrête toujours* si pour tout $w \in \Sigma^*$, il n'existe pas de suite infinie de transitions à partir de la configuration initiale $(q_{\text{init}}, w \sqcup \sqcup \cdots, 1)$. Le langage *reconnu* par $\mathcal{M}$ est l'ensemble des mots $w \in \Sigma^*$ qui peuvent mener à q_{acc}, c.-à-d.

$$\mathsf{Lang}(\mathcal{M}) \stackrel{\text{déf}}{=} \{w \in \Sigma^* : \exists u \in \Gamma^\omega, i \in \mathbb{N}_{>0} \text{ t.q. } (q_{\text{init}}, w \sqcup \sqcup \cdots, 1) \to^* (q_{\text{acc}}, u, i)\} \ .$$

Lorsque $\mathcal{M}$ s'arrête toujours, nous disons qu'elle *décide* $\mathsf{Lang}(\mathcal{M})$. Une machine de Turing déterministe $\mathcal{M}$ *calcule* une fonction $f : \Sigma^* \to \Sigma^*$ si $\mathcal{M}$ s'arrête toujours, et si pour tout $w \in \Sigma^*$, il existe $i \in \mathbb{N}_{>0}$ tel que

$$(q_{\text{init}}, w \sqcup \sqcup \cdots, 1) \to^* (q_{\text{acc}}, f(w) \sqcup \sqcup \cdots, i) \ .$$

Ces concepts nous permettent de formaliser la résolubilité d'un problème par un ordinateur :

Definition 2.2. Soient Σ un alphabet fini, $A \subseteq \Sigma^*$ et $f : \Sigma^* \to \Sigma^*$. Nous disons que le langage A est *décidable* s'il existe une machine de Turing $\mathcal{M}$ qui décide A, autrement, nous disons que A est *indécidable*. Nous disons que f est *calculable* s'il existe une machine de Turing déterministe $\mathcal{M}$ qui calcule f.

L'ensemble des machines de Turing est dénombrable puisque chaque machine de Turing peut être encodée naturellement par un mot de $\{0,1\}^*$. En particulier, nous dénotons Turing_i la $i^{\text{ème}}$ machine de Turing dans l'ordre lexicographique. Il est bien connu que le *problème d'arrêt* qui consiste à déterminer si une machine de Turing s'arrête sur sa propre représentation, c.-à-d. son encodage, est indécidable. Plus formellement, le langage suivant est indécidable :

$$\{i \in \mathbb{N}_{>0} \text{ encodé sur } \{0,1\}^* : \mathsf{Turing}_i \text{ s'arrête à partir de son encodage}\} .$$

Les réductions s'avèrent pratiques afin de montrer que d'autres langages sont indécidables :

Definition 2.3. Soient Σ un alphabet fini et $A, B \subseteq \Sigma^*$. Nous disons que A se *réduit, de façon multivoque, à* B, dénoté $A \leq_m B$ s'il existe une fonction calculable f telle que $w \in A \iff f(w) \in B$. Nous disons que A se *réduit, de façon Turing, à* B, dénoté $A \leq_T B$, s'il existe une machine de Turing avec oracle[2] pour B qui décide A. Lorsque $A \leq B$ et $B \leq A$ pour $\leq \in \{\leq_m, \leq_T\}$, nous disons que A et B sont *équivalents* sous la réduction $\leq$.

Proposition 2.4. *Soient Σ un alphabet fini, $A, B \subseteq \Sigma^*$ et $\leq \in \{\leq_m, \leq_T\}$. Si B est décidable et $A \leq B$, alors A est décidable. Si A est indécidable et $A \leq B$, alors B est indécidable.*

[2]Posséder un oracle pour B signifie que la machine possède une instruction spéciale qui permet de vérifier en temps constant si un mot $u \in \Sigma^*$ écrit sur un deuxième ruban appartient à B. Après un appel à l'oracle, le contenu du second ruban est effacé. Voir [Sip06, Per14] pour une définition formelle.

2.7.3 Complexité en temps

Nous introduisons une notion de temps d'exécution des machines de Turing, ce qui nous permettra de classifier les problèmes selon leur complexité calculatoire en *temps*.

Definition 2.5. Soient $\mathcal{M}$ une machine de Turing et $t : \mathbb{N} \to \mathbb{N}$. Nous disons que $\mathcal{M}$ *fonctionne en temps* t si sur toute entrée $w \in \Sigma^*$, $\mathcal{M}$ ne peut franchir plus de $t(|w|)$ transitions. Par extension, un langage A (resp. une fonction f) est *décidable (resp. calculable) en temps* t s'il existe une machine de Turing qui décide A (resp. calcule f) et fonctionne en temps t.

Les langages décidables peuvent être classifiés par leur complexité en temps. Nous rappelons les classes génériques usuelles :

$$\mathsf{NTIME}(f) \stackrel{\text{déf}}{=} \{A \subseteq \Sigma^* : A \text{ est décidé en temps } f' \in O(f)\} \text{ , et}$$

$$\mathsf{DTIME}(f) \stackrel{\text{déf}}{=} \{A \subseteq \Sigma^* : A \text{ est décidé en temps } f' \in O(f) \text{ par une machine de Turing déterministe}\} \text{ .}$$

Les classes concrètes qui seront considérées dans cette thèse sont les suivantes :

$$\mathsf{P} \stackrel{\text{déf}}{=} \bigcup_{k\in\mathbb{N}} \mathsf{DTIME}(n^k) \text{ , } \qquad \mathsf{NP} \stackrel{\text{déf}}{=} \bigcup_{k\in\mathbb{N}} \mathsf{NTIME}(n^k) \text{ ,}$$

$$\mathsf{EXPTIME} \stackrel{\text{déf}}{=} \bigcup_{k\in\mathbb{N}} \mathsf{DTIME}(2^{n^k}) \text{ , } \qquad \mathsf{NEXPTIME} \stackrel{\text{déf}}{=} \bigcup_{k\in\mathbb{N}} \mathsf{NTIME}(2^{n^k}) \text{ ,}$$

$$2\text{-}\mathsf{EXPTIME} \stackrel{\text{déf}}{=} \bigcup_{k\in\mathbb{N}} \mathsf{DTIME}(2^{2^{n^k}}) \text{ , et } \quad 2\text{-}\mathsf{NEXPTIME} \stackrel{\text{déf}}{=} \bigcup_{k\in\mathbb{N}} \mathsf{NTIME}(2^{2^{n^k}}) \text{ .}$$

2.7.4 Complexité en espace

Nous introduisons une notion de mémoire utilisée par les machines de Turing afin de classifier les problèmes selon leur complexité calculatoire en *espace*.

Definition 2.6. Soient $\mathcal{M}$ une machine de Turing et $\ell : \mathbb{N} \to \mathbb{N}$. Nous disons que

$\mathcal{M}$ *fonctionne en espace* ℓ si sur toute entrée $w \in \Sigma^*$, $\mathcal{M}$ n'occupe à aucun moment plus de $\ell(|w|)$ cases de son ruban. Par extension, un langage A (resp. une fonction f) est *décidable (resp. calculable) en espace* ℓ s'il existe une machine de Turing qui décide A (resp. calcule f) et fonctionne en espace ℓ.

Les langages décidables peuvent également être classifiés selon leur complexité en espace grâce aux deux classes génériques suivantes :

$$\mathsf{NSPACE}(f) \stackrel{\text{déf}}{=} \{A \subseteq \Sigma^* : A \text{ est décidé en espace } f' \in O(f)\} \ ,$$

$$\mathsf{DSPACE}(f) \stackrel{\text{déf}}{=} \{A \subseteq \Sigma^* : A \text{ est décidé en espace } f' \in O(f) \text{ par une machine de Turing déterministe}\} \ .$$

Les classes concrètes qui seront considérées dans cette thèse sont les suivantes :

$$\mathsf{L} \stackrel{\text{déf}}{=} \mathsf{DSPACE}(\log(n)) \ , \qquad \mathsf{NL} \stackrel{\text{déf}}{=} \mathsf{NSPACE}(\log(n)) \ ,$$

$$\mathsf{PSPACE} \stackrel{\text{déf}}{=} \bigcup_{k\in\mathbb{N}} \mathsf{DSPACE}(n^k) \ , \qquad \mathsf{NPSPACE} \stackrel{\text{déf}}{=} \bigcup_{k\in\mathbb{N}} \mathsf{NSPACE}(n^k) \ ,$$

$$\mathsf{EXPSPACE} \stackrel{\text{déf}}{=} \bigcup_{k\in\mathbb{N}} \mathsf{DSPACE}(2^{n^k}) \ , \text{ et } \quad \mathsf{NEXPSPACE} \stackrel{\text{déf}}{=} \bigcup_{k\in\mathbb{N}} \mathsf{NSPACE}(2^{n^k}) \ .$$

Les classes déterministes et non déterministes sont reliées ainsi :

Théorème de Savitch. Pour toute fonction $f(n) \geq \log n$,

$$\mathsf{NSPACE}(f) \subseteq \mathsf{DSPACE}(f^2) \subseteq \mathsf{NSPACE}(f^2) \ .$$

Certaines inclusions connues entre les différentes classes de complexité introduites sont illustrées à la figure 2.2.

2.7.5 Réductions et complétude

Afin d'identifier les problèmes les « plus difficiles » d'une classe de complexité, nous ferons usage du concept standard de réductions. Nous nous limitons ici aux

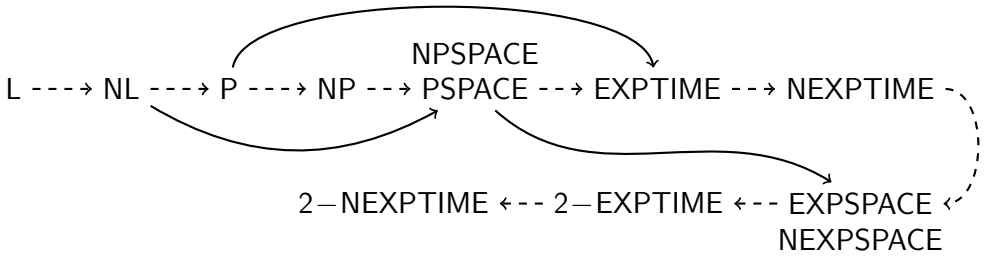


Figure 2.2 – Certaines inclusions connues entre les différentes classes de complexité. Les arcs tiretés indiquent des inclusions non strictes $\subseteq$, et les arcs solides indiquent des inclusions strictes $\subset$.

réductions pouvant être calculées en espace logaritmique.

Definition 2.7. Soient Σ un alphabet fini et $A, B \subseteq \Sigma^*$. Nous écrivons $A \leq_m^{\log} B$ s'il existe une fonction $f : \Sigma^* \to \Sigma^*$ calculable[3] en espace $O(\log n)$ telle que $w \in A \iff f(w) \in B$. Nous écrivons $A \leq_T^{\log} B$ s'il existe une machine de Turing avec oracle pour B qui décide A en espace logarithmique[4]. Lorsque $A \leq B$ et $B \leq A$ pour $\leq \in \{\leq_m^{\log}, \leq_T^{\log}\}$, nous disons que A et B sont *équivalents* sous la réduction $\leq$.

Definition 2.8. Soit $\mathcal{C}$ l'une des classes de complexité définies plus tôt (excluant L). Nous disons qu'un langage A est $\mathcal{C}$-*difficile* si $A \leq_m^{\log} B$ pour tout $B \in \mathcal{C}$. Si de plus $A \in \mathcal{C}$, nous disons que A est $\mathcal{C}$-*complet*. Lorsque $\leq_m^{\log}$ est substituée par $\leq_T^{\log}$, nous spécifions « sous réduction Turing ».

2.8 Systèmes de transitions

Dans ce chapitre, nous introduisons les systèmes de transitions qui seront étudiés dans l'entiereté de cette thèse. Les systèmes de transitions forment un modèle abstrait qui permet notamment de décrire le comportement de modèles de calcul.

[3] Afin de permettre à la machine de Turing qui calcule f de lire toute son entrée, nous supposons qu'elle possède deux rubans. Le premier ruban ne permet que de lire l'entrée, alors que le second ruban permet la lecture et l'écriture, mais ne peut excéder $O(\log n)$ cases.

[4] De manière similaire à $\leq_m^{\log}$, la machine de Turing possède un ruban d'entrée et un ruban de travail.

Un système de transitions est constitué d'un ensemble, normalement infini, d'états, et de relations de transition qui indiquent quels états peuvent mener à quels états. Ainsi, un système de transitions, dans sa forme la plus générale, peut être représenté par un graphe infini dont les arcs sont étiquetés par les lettres d'un alphabet.

Les modèles de calcul utilisés en informatique théorique induisent tous au moins un système de transitions, p. ex. rappelons qu'une machine de Turing induit un système de transitions où les états sont l'ensemble des configurations, et où l'unique relation de transitions indique vers quelle configuration une configuration peut mener en une étape de la machine de Turing.

Formellement, les systèmes de transitions sont définis de la façon suivante :

Definition 2.9. Un *système de transitions (étiqueté)* $\mathcal{S} = (X, \to, \Sigma)$ est un ensemble X muni de relations étiquetées par Σ ; plus formellement,

- X est un ensemble dénombrable,
- Σ est un ensemble dénombrable non vide,
- $\xrightarrow{\sigma} \subseteq X \times X$ pour chaque $\sigma \in \Sigma$.

Les éléments de X sont appelés les *états* ou *configurations* de $\mathcal{S}$ et les relations $\xrightarrow{\sigma}$ sont appelées les *relations de transition* sur les états. Bien que $\to$ soit une collection de relations, nous utiliserons également $\to$ afin de représenter la relation $\to \stackrel{\text{déf}}{=} \bigcup_{\sigma\in\Sigma} \xrightarrow{\sigma}$; cette relation fait ainsi abstraction des étiquettes. Bien que le nombre de relations de transition puisse être infini, la majorité des systèmes de transitions qui seront étudiés en possèdent un nombre fini, et souvent même une seule. Ainsi, lorsque Σ est un singleton $\{a\}$, nous écrivons simplement $\mathcal{S} = (X, \to)$ où $\to$ représente l'unique relation de transitions $\xrightarrow{a}$, ce qui concorde avec notre convention $\to = \bigcup_{\sigma\in\{a\}} \xrightarrow{\sigma}$. Un système de transitions $\mathcal{S} = (X, \to, \Sigma)$ équipé d'un préordre $\leq \subseteq X \times X$ est appelé *système de transitions ordonné* et est dénoté $\mathcal{S} = (X, \to, \Sigma, \leq)$, ou simplement $\mathcal{S} = (X, \to, \leq)$ lorsque $|\Sigma| = 1$.

Nous introduisons un type particulier de systèmes de transitions qui apparaît souvent dans la littérature et que nous étudierons directement ou indirectement

dans tous les chapitres subséquents.

Definition 2.10. Un système de transitions *fonctionnel* est un système de transitions $\mathcal{S} = (X, \to, F)$ tel que ses relations de transition sont définies par un ensemble F de fonctions partielles de X vers X. Plus précisément, $x \xrightarrow{f} y$ si, et seulement si, $f(x) = y$.

2.8.1 Relations de transition

Soient $\sigma_1, \sigma_2, \ldots, \sigma_m \in \Sigma$ et $\pi, \pi' \in \Sigma^*$, nous étendons les transitions aux séquences finies d'étiquettes, c.-à-d. aux mots finis sur Σ, de la façon suivante :

$$\begin{aligned} \xrightarrow{\varepsilon} &\stackrel{\text{déf}}{=} \mathsf{Id}\ , \\ \xrightarrow{\sigma_1\sigma_2\cdots\sigma_m} &\stackrel{\text{déf}}{=} \xrightarrow{\sigma_m} \circ \ldots \circ \xrightarrow{\sigma_2} \circ \xrightarrow{\sigma_1}\ , \text{ et} \\ \xrightarrow{\pi\pi'} &\stackrel{\text{déf}}{=} \xrightarrow{\pi'} \circ \xrightarrow{\pi}\ . \end{aligned} \tag{2.1}$$

Soit $\pi \in \Sigma^*$, nous étendons la relation $\xrightarrow{\pi}$ à plusieurs répétitions :

$$\xrightarrow{\pi}{}^{k} \stackrel{\text{déf}}{=} \underbrace{\xrightarrow{\pi} \circ \xrightarrow{\pi} \circ \cdots \circ \xrightarrow{\pi}}_{k \text{ fois}}\ , \quad \xrightarrow{\pi}{}^{+} \stackrel{\text{déf}}{=} \bigcup_{k \geq 1} \xrightarrow{\pi}{}^{k} \quad \text{et} \quad \xrightarrow{\pi}{}^{*} \stackrel{\text{déf}}{=} \xrightarrow{\varepsilon} \cup \xrightarrow{\pi}{}^{+}\ . \tag{2.2}$$

En d'autres termes, $\xrightarrow{\pi}{}^{+}$ est la fermeture transitive de $\xrightarrow{\pi}$, et $\xrightarrow{\pi}{}^{*}$ est la fermeture réflexive et transitive de $\xrightarrow{\pi}$. Cette même notation s'applique à $\to$, c.-à-d. $\to^k$, $\to^+$ et $\to^*$ sont définis similairement. Notons que la relation $\xrightarrow{\pi}{}^{k}$ est égale à $\xrightarrow{\pi^k}$, mais afin d'être consistants nous préférons la première écriture puisque la seconde ne s'applique pas dans le cas de $\to$.

Soient $\sigma \in \Sigma$, $x, y \in X$ et $A \subseteq X$, nous écrivons $x \xrightarrow{\sigma}_A y$ si $x \xrightarrow{\sigma} y$ et $x, y \in A$. Nous étendons cette notation aux séquences et aux répétitions de la même façon qu'en (2.1) et (2.2) en remplaçant simplement $\xrightarrow{\sigma}$ par $\xrightarrow{\sigma}_A$. Cette notation s'applique également à $\to$ sans égard aux étiquettes.

2.8.2 Ensembles d'accessibilité

Nous définissons, respectivement, l'ensemble des *prédecesseurs immédiats* et des *successeurs immédiats* de $x \in X$ par

$$\begin{aligned} \mathsf{Pred}(x) &\stackrel{\text{déf}}{=} \{y \in X : y \to x\} \text{ , et} \\ \mathsf{Succ}(x) &\stackrel{\text{déf}}{=} \{y \in X : x \to y\} \ . \end{aligned}$$

Les ensembles Pred et Succ sont naturellement étendus à un ensemble d'états $B \subseteq X$:

$$\mathsf{Pred}(B) \stackrel{\text{déf}}{=} \bigcup_{x \in B} \mathsf{Pred}(x) \quad \text{et} \quad \mathsf{Succ}(B) \stackrel{\text{déf}}{=} \bigcup_{x \in B} \mathsf{Succ}(x) \ .$$

Nous définissons également l'ensemble des *prédecesseurs* et des *successeurs* de $x \in X$ par

$$\begin{aligned} \mathsf{Pred}^*(x) &\stackrel{\text{déf}}{=} \{y \in X : y \to^* x\} \text{ , et} \\ \mathsf{Succ}^*(x) &\stackrel{\text{déf}}{=} \{y \in X : x \to^* y\} \ . \end{aligned}$$

Ces ensembles sont également étendus à un ensemble d'états $B \subseteq X$:

$$\mathsf{Pred}^*(B) \stackrel{\text{déf}}{=} \bigcup_{x \in B} \mathsf{Pred}^*(x) \quad \text{et} \quad \mathsf{Succ}^*(B) \stackrel{\text{déf}}{=} \bigcup_{x \in B} \mathsf{Succ}^*(x) \ .$$

2.8.3 Chemins et exécutions

Soit $A \subseteq X$, nous disons que $\pi \in \Sigma^*$ est un *chemin, restreint à A, ...*

(i) *...* , si, et seulement si, $x \xrightarrow{\pi}_A y$ pour certains $x, y \in X$,

(ii) *... à partir de x*, si, et seulement si, $x \xrightarrow{\pi}_A y$ pour un certain $y \in X$,

(iii) *... vers y* , si, et seulement si, $x \xrightarrow{\pi}_A y$ pour un certain $x \in X$,

(iv) *... de x vers y* , si, et seulement si, $x \xrightarrow{\pi}_A y$.

Notons que le vocabulaire des points (iii) et (iv) ne s'étend pas aux séquences infinies $\pi \in \Sigma^\omega$ puisque le dernier état y n'existe pas. Néanmoins, nous pouvons étendre les points (i) et (ii) aux séquences infinies. Soit $\pi \in \Sigma^\omega$, nous disons que π est (a) un *chemin, restreint à* A, si, et seulement si, il existe $x_0, x_1, \ldots \in X$ tels que $x_0 \xrightarrow{\pi_1}_A x_1 \xrightarrow{\pi_2}_A \ldots$; (b) un *chemin, restreint à* A*, à partir de* x, si, et seulement si, il existe $x_1, x_2, \ldots \in X$ tels que $x \xrightarrow{\pi_1}_A x_1 \xrightarrow{\pi_2}_A x_2 \ldots$.

Soient $A \subseteq X$ et $x \in X$, nous définissons respectivement $\mathsf{Chemins}^{\mathsf{fin}}_A(x)$, $\mathsf{Chemins}^{\mathsf{inf}}_A(x)$ et $\mathsf{Chemins}_A(x)$, comme étant l'ensemble des chemins finis, l'ensemble des chemins infinis, et l'ensemble des chemins, restreints à A, à partir de x :

$$\begin{aligned}
\mathsf{Chemins}^{\mathsf{fin}}_A(x) &\stackrel{\text{déf}}{=} \{\pi \in \Sigma^* : \pi \text{ est un chemin, resteint à } A, \text{ à partir de } x\} \text{ , et} \\
\mathsf{Chemins}^{\mathsf{inf}}_A(x) &\stackrel{\text{déf}}{=} \{\pi \in \Sigma^\omega : \pi \text{ est un chemin, resteint à } A, \text{ à partir de } x\} \text{ ,} \\
\mathsf{Chemins}_A(x) &\stackrel{\text{déf}}{=} \mathsf{Chemins}^{\mathsf{fin}}_A(x) \cup \mathsf{Chemins}^{\mathsf{inf}}_A(x) \text{ .}
\end{aligned}$$

Une *exécution*, restreinte à $A \subseteq X$, est une séquence d'états $x \in A^\infty$ pour laquelle il existe un chemin $\pi \in \Sigma^\infty$ tel que, pour tout $1 \leq i < |x|$, $x_i \xrightarrow{\pi_i}_A x_{i+1}$. Soit x une exécution non vide, nous appelons son premier état x_1 l'*état initial*, et si x est fini, nous appelons son dernier état $x_{|x|}$ l'*état final*.

Soient $A \subseteq X$ et $x \in X$, de façon analogue aux chemins, nous définissons respectivement $\mathsf{Exec}^{\mathsf{fin}}_A(x), \mathsf{Exec}^{\mathsf{inf}}_A(x)$ et $\mathsf{Exec}_A(x)$, comme étant l'ensemble des exécutions finies, l'ensemble des exécutions infinies, et l'ensemble des exécutions, restreintes à A, à partir de x :

$$\begin{aligned}
\mathsf{Exec}^{\mathsf{fin}}_A(x) &\stackrel{\text{déf}}{=} \{y \in A^* : y \text{ est une exécution avec état initial } x\} \text{ , et} \\
\mathsf{Exec}^{\mathsf{inf}}_A(x) &\stackrel{\text{déf}}{=} \{y \in A^\omega : y \text{ est une exécution avec état initial } x\} \text{ ,} \\
\mathsf{Exec}_A(x) &\stackrel{\text{déf}}{=} \mathsf{Exec}^{\mathsf{fin}}_A(x) \cup \mathsf{Exec}^{\mathsf{inf}}_A(x) \text{ .}
\end{aligned}$$

Nous dénotons respectivement par $\mathsf{ExecMax}^{\mathsf{fin}}_A(x)$ et $\mathsf{ExecMax}_A(x)$ l'ensemble des exécutions finies maximales et l'ensemble des exécutions maximales, restreintes à

A, à partir de x :

$$\mathsf{ExecMax}^{\mathsf{fin}}_A(x) \stackrel{\text{déf}}{=} \{y \in \mathsf{Exec}^{\mathsf{fin}}_A(x) : \mathsf{Succ}(y_{|y|}) = \emptyset\} \text{ , et}$$
$$\mathsf{ExecMax}_A(x) \stackrel{\text{déf}}{=} \mathsf{ExecMax}^{\mathsf{fin}}_A(x) \cup \mathsf{Exec}^{\mathsf{inf}}_A(x) \text{ .}$$

Pour toutes ces notations, lorsque $A = X$, c.-à-d. lorsque A est l'ensemble de tous les états, nous omettons l'indice A, p. ex. $\mathsf{Chemins}_X(x)$ s'écrit simplement $\mathsf{Chemins}(x)$. De plus, nous ne spécifions pas « restreint à A » lorsque $A = X$. Nous étendons également ces notations aux ensembles d'états, p. ex. pour $B \subseteq X$, nous écrivons $\mathsf{ExecMax}(B)$ pour dénoter $\bigcup_{x \in B} \mathsf{ExecMax}(x)$.

2.8.4 Branchement, classes et effectivité

Afin de paramétrer les différents problèmes de décision qui seront étudiés dans cette thèse, nous introduisons la notion de classe de systèmes de transitions.

Definition 2.11. Une *classe $\mathcal{C}$ de systèmes de transitions* est un ensemble dénombrable de systèmes de transitions. Nous dénotons le $i^{\text{ème}}$ système de transitions de $\mathcal{C}$ par $\mathcal{C}(i)$.

Afin de pouvoir manipuler les systèmes de transitions, nous étendons les définitions utilisées par Finkel & Goubault-Larrecq [FG12]. Soit $\mathcal{C}$ une classe de systèmes de transitions telle que X_i est l'ensemble des états de $\mathcal{C}(i)$. Nous associons à $\mathcal{C}$ une fonction surjective $\mathsf{repr}_{\mathcal{C}} : \mathbb{N} \times \mathbb{N} \to \bigcup_i X_i$ telle que l'ensemble $\{(i, e) : \mathsf{repr}_{\mathcal{C}}(i, e) \in X_i\}$ soit décidable. Nous définissons $\mathsf{enc}_{\mathcal{C}} : \mathbb{N} \to 2^{\mathbb{N}}$ par

$$\mathsf{enc}_{\mathcal{C}}(i) \stackrel{\text{déf}}{=} \{e \in \mathbb{N} : \mathsf{repr}_{\mathcal{C}}(i, e) \in X_i\} \text{ .}$$

Nous disons que $e \in \mathbb{N}$ est un *encodage* de $x \in X_i$ si $\mathsf{repr}_{\mathcal{C}}(i, e) = x$. Ainsi, l'ensemble $\mathsf{enc}_{\mathcal{C}}(i)$ encode X_i. Nous disons qu'une machine de Turing M travaillant sur $\mathbb{N} \times \mathbb{N}$ *calcule* une relation $\rho \subseteq X_i \times X_i$ si

- M s'arrête, au moins, sur $\mathsf{enc}_{\mathcal{C}}(i) \times \mathsf{enc}_{\mathcal{C}}(i)$, et

- pour tout $(e, e') \in \mathsf{enc}_{\mathcal{C}}(i) \times \mathsf{enc}_{\mathcal{C}}(i)$, M accepte (e, e') si, et seulement si, $(\mathsf{repr}_{\mathcal{C}}(i, e), \mathsf{repr}_{\mathcal{C}}(i, e')) \in \rho$.

Dans cette thèse, nous supposerons que les systèmes de transitions sont effectifs au sens suivant :

Definition 2.12. Une classe $\mathcal{C}$ de systèmes de transitions est dite *effective* s'il existe une machines de Turing $M_{\rightarrow}$, travaillant sur $\mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}$, telle pour tout $i \in \mathbb{N}$, $M_{\rightarrow}$ calcule la relation de transitions $\xrightarrow{\sigma}$ de $\mathcal{C}(i) = (X_i, \rightarrow, \Sigma)$ lorsque son premier argument est fixé à i et lorsque son deuxième argument est fixé à j où σ est la $j^{\text{ème}}$ étiquette de Σ.

Additionnellement, si $\mathcal{C}$ est une classe de systèmes de transitions ordonnés, nous exigeons qu'il existe une seconde machine $M_{\leq}$, travaillant sur $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$, telle pour tout $i \in \mathbb{N}$, $M_{\leq}$ calcule l'ordre $\leq$ de $\mathcal{C}(i)$ lorsque son premier argument est fixé à i.

Par extension, nous disons qu'un système de transitions $\mathcal{S}$ est *effectif* lorsque la classe dégénérée $\{\mathcal{S}\}$ est effective.

Notons qu'il peut y avoir une distinction d'uniformité subtile, mais importante, entre une classe effective de systèmes de transitions et une classe de systèmes de transitions dont chacun de ses systèmes de transitions est individuellement effectif :

Proposition 2.13. *Il existe une classe $\mathcal{C}$ de systèmes de transitions qui n'est pas effective bien que chaque $\mathcal{S} \in \mathcal{C}$ soit effectif.*

Démonstration. Soient $i \in \mathbb{N}$, et $\mathcal{S}_i = (\{0\}, \rightarrow)$ le système de transitions tel que

$$\rightarrow \stackrel{\text{déf}}{=} \begin{cases} \{(0,0)\} & \text{si } \mathsf{Turing}_i \text{ s'arrête sur son encodage,} \\ \emptyset & \text{sinon.} \end{cases}$$

$\mathcal{S}_i$ est effectif car il existe une machine de Turing qui calcule $\rightarrow$, c.-à-d., ou bien la machine qui retourne toujours vrai, ou bien la machine qui retourne toujours faux.

Soit $\mathcal{C} = \{S_i : i \in \mathbb{N}\}$. Supposons que $\mathcal{C}$ soit une classe effective de systèmes de transitions, alors il existe une machine de Turing qui calcule la relation de

transitions de S_i lorsque son premier argument est fixé à i. Ainsi, il est possible de décider si Turing_i s'arrête sur son encodage en vérifiant si $0 \to 0$ dans S_i. Ceci est une contradiction, donc $\mathcal{C}$ n'est pas effective. □

Afin d'analyser les systèmes de transitions, il est souvent supposé dans la littérature qu'il est possible d'obtenir les successeurs immédiats $\mathsf{Succ}(x)$ pour tout état x. Or, cela ne fait du sens que lorsque $\mathsf{Succ}(x)$ est fini pour tout état x, et cela ne sera pas toujours le cas dans cette thèse puisque les systèmes de transitions sont introduits dans leur pleine généralité. Nous introduisons donc la notion de branchement.

Definition 2.14. Un système de transitions $\mathcal{S} = (X, \to, \Sigma)$ est dit *à branchement fini* lorsque $\mathsf{Succ}(x)$ est fini pour tout $x \in X$. Dans le cas contraire, $\mathcal{S}$ est dit *à branchement infini*.

Nous adaptons la notion d'effectivité qui permet de calculer $\mathsf{Succ}(x)$ aux systèmes de transitions à branchement infini en étendant la notion d'effectivité présentée à la définition 2.12 :

Definition 2.15. Une classe $\mathcal{C}$ de systèmes de transitions est dite *post-effective*[5] si elle est effective, et s'il existe additionnellement une machine de Turing $M_{|\mathsf{Succ}|}$ qui calcule $|\mathsf{Succ}_{\mathcal{C}(i)}(x)| \in \mathbb{N} \cup \{\infty\}$ sur entrée (i, x), où $i \in \mathbb{N}$ et x est un état de $\mathcal{C}(i)$[6]. Par extension, nous disons qu'un système de transitions $\mathcal{S}$ est *post-effectif* lorsque la classe dégénérée $\{\mathcal{S}\}$ est post-effective.

Lorsqu'une classe $\mathcal{C}$ de systèmes de transitions est post-effective, il est possible d'énumérer les éléments de $\mathsf{Succ}_{\mathcal{C}(i)}(x)$ pour tout état x, et en particulier d'obtenir tous les éléments de $\mathsf{Succ}_{\mathcal{C}(i)}(x)$ lorsque celui-ci est fini. En effet, il suffit d'exécuter $M_\to$ sur (i, j, x, y) pour toute étiquette j et pour tout état y et d'arrêter lorsque le nombre de successeurs y trouvés est égal à la cardinalité de $\mathsf{Succ}_{\mathcal{C}(i)}(x)$ donnée par $M_{|\mathsf{Succ}|}$.

[5] Nous choisissons l'appellation « post-effective » plutôt que « succ-effective » afin d'être consistent avec [BFM14, BFM16].

[6] Comme à la définition 2.12, les états de $\mathcal{C}(i)$ sont manipulés à l'aide de $\mathsf{repr}_{\mathcal{C}}$, et $M_{|\mathsf{Succ}|}$ s'arrête, au moins, sur $\mathbb{N} \times \mathsf{enc}_{\mathcal{C}}(i)$.

2.8.5 Problèmes de décision

Le but principal de cette thèse est d'étudier la complexité et la décidabilité des différents problèmes de décision de systèmes de transitions. Nous les introduisons ici dans leur forme la plus générale comme référence pour le reste du document. Soit $\mathcal{C}$ une classe de systèmes de transitions, nous paramétrons ces problèmes par $\mathcal{C}$, c.-à-d., les systèmes de transitions sont donnés en entrée par leur indice $i \in \mathbb{N}$ dans $\mathcal{C}$.

Accessibilité

Entrée: Un système de transitions $\mathcal{S} = (X, \to, \Sigma)$ et $x, y \in X$.
Question: $x \to^* y$?

Terminaison

Entrée: Un système de transitions $\mathcal{S} = (X, \to, \Sigma)$ et $x \in X$.
Question: $\mathsf{Exec}^{\mathsf{inf}}(x) = \emptyset$?

Terminaison forte

Entrée: Un système de transitions $\mathcal{S} = (X, \to, \Sigma)$ et $x \in X$.
Question: Existe-t-il $m \in \mathbb{N}$ tel que $y \in \mathsf{Exec}(x) \implies |y| \leq m$?

Finitude

Entrée: Un système de transitions $\mathcal{S} = (X, \to, \Sigma)$ et $x \in X$
Question: Existe-t-il $m \in \mathbb{N}$ tel que $|\mathsf{Succ}^*(x)| \leq m$?

Maintenabilité

Entrée: Un système de transitions ordonné $\mathcal{S} = (X, \to, \Sigma, \leq)$, $x \in X$ et a_1, $a_2, \ldots, a_n \in X$.
Question: Soit $A = \uparrow\{a_1, a_2, \ldots, a_n\}$. Est-ce que $\mathsf{ExecMax}_A(x) \neq \emptyset$?

MAINTENABILITÉ FAIBLE

Entrée: Un système de transitions ordonné $\mathcal{S} = (X, \to, \Sigma, \leq)$, $x \in X$ et a_1, $a_2, \ldots, a_n \in X$.

Question: Soit $A = \uparrow\{a_1, a_2, \ldots, a_n\}$. Est-ce que pour tout $m \in \mathbb{N}$, il existe $y \in \mathsf{Exec}_A(x)$ tel que $|y| \geq m$?

COUVERTURE

Entrée: Un système de transitions ordonné $\mathcal{S} = (X, \to, \Sigma, \leq)$ et $x, y \in X$.

Question: Existe-t-il $y' \in \uparrow y$ tel que $x \to^* y'$?

2.8.6 Systèmes de transitions bien structurés

Tel qu'indiqué plus tôt, notre étude concerne les problèmes de décisions présentés à la section 2.8.5. Ces problèmes sont tous indécidables pour les classes de systèmes de transitions les plus générales. Afin de rendre leur étude plus intéressante et de mieux représenter les modèles concrets, une approche adoptée dans la littérature consiste à étudier les systèmes de transitions ordonnés dont l'ordre est un beau préordre muni d'une certaine forme de monotonicité. Ces systèmes sont appelés systèmes de transitions bien structurés et ont d'abord été introduits par Finkel [Fin87a, Fin87b, Fin90], puis étudiés par Abdulla *et al.* [ACJT96, ACJT00], sous plusieurs définitions plus ou moins similaires finalement uniformisées par Finkel & Schnoebelen [FPS01]. Les systèmes de transitions bien structurés englobent de nombreux modèles concrets dont la plupart des machines à compteurs considérées dans cette thèse. L'étude des systèmes de transitions bien structurés permet d'étudier génériquement la décidabilité des différents problèmes de décision sans se soucier de leur complexité calculatoire, et des détails spécifiques à chaque modèle concret.

Les systèmes de transitions bien structurés sont définis formellement de la façon suivante :

Definition 2.16 ([Fin87a, Fin87b, Fin90])**.** Un *système de transitions bien struc-*

turé, abrégé *WSTS*, est un système de transitions ordonné[7] $\mathcal{S} = (X, \to, \Sigma, \leq)$ tel que $\leq$ est un beau préordre et $\to$ satisfait la règle de *monotonicité* suivante :

$$\forall x, x', y \in X \quad x \leq x' \text{ et } x \to y \implies \exists y' \in X \text{ t.q. } y \leq y' \text{ et } x' \to^* y' \ .$$

Il existe plusieurs variantes de monotonicité, dans la littérature, qui diffèrent de la monotonicité, dite *standard*, présentée à la définition 2.16. Nous en présentons quelques-unes tirées de [FPS01] qui seront utilisées dans cette thèse, en illustrant en couleur ( ) leurs différences avec la monotonicité standard :

$$\begin{array}{lllll}
\textit{forte :} & x \leq x' \text{ et } x \to y & \implies \exists y' & \text{t.q. } y \leq y' \text{ et } x' \to & y' \ . \\
\textit{bégayante}^{8} \textit{ :} & x \leq x' \text{ et } x \to y & \implies \exists x'', y' & \text{t.q. } y \leq y' \text{ et } x' \to^*_{\uparrow x} x'' \to & y' \ . \\
\textit{transitive :} & x \leq x' \text{ et } x \to y & \implies \exists y' & \text{t.q. } y \leq y' \text{ et } x' \to^+ & y' \ . \\
\textit{stricte :} & x < x' \text{ et } x \to y & \implies \exists y' & \text{t.q. } y < y' \text{ et } x' \to^* & y' \ .
\end{array}$$

Notons que la monotonicité forte est un cas particulier de la monotonicité bégayante, qui est à son tour un cas particulier de la monotonicité transitive, qui est elle-même un cas particulier de la monotonicité standard. La monotonicité stricte est incomparable avec ces autres types de monotonicité.

Nous remarquons que les systèmes de transitions fonctionnels forment une classe de WSTS lorsque munis d'un beau préordre et lorsque leurs fonctions sont non décroissantes.

Proposition 2.17. *Soit $\mathcal{S} = (X, \to, F, \leq)$ un système de transitions fonctionnel et ordonné. Si $\leq$ est un beau préodre pour X, et que F est un ensemble de fonctions partielles non décroissantes (resp. croissantes), alors $\mathcal{S}$ est un WSTS avec monotonicité forte (resp. forte et stricte).*

Démonstration. Soit $\mathcal{S}$ tel que décrit dans l'énoncé de la proposition. Puisque $\leq$ est un beau préordre, il suffit de montrer que $\to$ est monotone. Soient $x, y, x' \in X$

[7] La définition donnée par [FPS01] ne concerne pas les systèmes de transitions étiquetés. Nous les incluons ici, sans que cela ne cause problème, afin de s'adapter à l'ensemble des modèles qui seront étudiés.

[8] Traduction libre du terme anglais « stuttering monotonicity ».

tels que $x < x'$ et $x \to y$. Par définition de $\to$, $f(x) = y$ pour un certain $f \in F$. Ainsi, puisque f est non décroissante (resp. croissante), $f(x')$ est définie, et par conséquent $y = f(x) \leq f(x') = y'$ (resp. $y = f(x) < f(x') = y'$) pour un certain $y' \in X$. Nous avons donc $x' \to y'$ et $y \leq y'$ (resp. $y \leq y'$). □

Nous notons également que la post-effectivité d'une classe de WSTS ne permet pas de déterminer si l'un de ses WSTS est à branchemement infini, même lorsque les WSTS possèdent les monotonicité les plus restreintes.

Proposition 2.18. *Il existe une classe $\mathcal{C}$ de WSTS avec monotonicité forte et stricte pour laquelle le problème qui consiste à tester, sur entrée $i \in \mathbb{N}$, si $\mathcal{C}(i)$ est à branchement infini est indécidable.*

Démonstration. Soient $i \in \mathbb{N}$, et $\mathcal{C}(i) \stackrel{\text{déf}}{=} (\mathbb{N}, \to, \leq)$ le système de transitions ordonné tel que $\to$ est défini par

$x \to x$	si Turing_i ne s'arrête pas sur son encodage en x étapes ou moins,
$x \to x, x+1, x+2, \ldots$	sinon.

Nous remarquons que $\mathcal{C}(i)$ possède la monotonicité forte et stricte. Soient $x, x', y \in \mathbb{N}$ tels que $x < x'$ et $x \to y$. Si $y = x$, alors $x' \to x'$. En effet, si $y > x$, alors Turing_i s'arrête en x étapes ou moins, et donc également en x' étapes ou moins. Ainsi, $x' \to y'$ pour tout $y' \geq x'$ et en donc pour tout $y' > y$.

Nous notons également que $\mathcal{C}$ est une classe effective de WSTS puisque $\to$ et $|\mathsf{Succ}_{\mathcal{C}(i)}(x)|$ peuvent être calculées en exécutant Turing_i pour un nombre fini d'étapes.

De plus, pour tout i, Turing_i s'arrête sur son encodage si, et seulement si, il existe $x \in \mathbb{N}$ tel que $\mathsf{Succ}_{\mathcal{C}(i)}(x)$ est infini. Ainsi, s'il était possible de tester si $\mathcal{C}(i)$ est à branchement infini, il serait possible de décider le problème d'arrêt, ce qui est impossible. □

2.9 Machines à compteurs

Une *machine à compteurs* est intuitivement un ensemble fini de compteurs, contenant chacun un nombre entier, et un ensemble fini d'instructions qui manipulent ces compteurs. Les compteurs sont souvent restreints aux entiers non négatifs. Parmi les instructions les plus fréquentes, nous retrouvons :

- l'*addition* : ajoute $n \in \mathbb{N}$ au compteur i,
- la *soustraction* : soustrait n du compteur i,
- la *soustraction conditionnelle* : soustrait $n \in \mathbb{N}$ du compteur i si sa valeur est plus grande ou égale à n,
- le *transfert* : copie le contenu du compteur i au compteur j,
- la *remise à zéro* : remplace la valeur du compteur i par 0, et
- le *test à zéro* : vérifie que la valeur du compteur i est 0.

Les machines à compteurs sont parfois représentées sous forme de programmes, c.-à-d. une liste d'instructions et de branchements conditionnels, et parfois sous forme d'automates finis où les transitions sont étiquetées par des instructions.
Par exemple, considérons la machine à compteurs décrites à l'algorithme 2.1. Celle-ci possède trois compteurs, x, y, z, qui contiennent initialement les valeurs entières non négatives a, b, c. À la fin de l'unique exécution de la machine, $x = y = 0$ et $z = a + 2b$. De façon équivalente, cette machine peut également être décrite par un automate, tel qu'illustré à la figure 2.3.
La machine décrite dans l'exemple précédent utilise le test à zéro. Or, en général, le test à zéro permet d'obtenir toute la puissance des machines de Turing. En effet, il est bien connu que les *machines de Minsky* à deux compteurs, c'est-à-dire les machines à deux compteurs non négatifs munis de l'addition, de la soustraction conditionnelle et du test à zéro, sont équivalentes aux machines de Turing en terme d'expressivité et de calculabilité (voir, p. ex., [HMU01, section 8.5.3]). De façon

Algorithme 2.1 : Exemple de machine à compteurs, sous forme de programme, calculant $x + 2y$.

Entrées : Variables $x, y, z \in \mathbb{N}$
Sorties : $z = x + 2y$

1 $z \leftarrow 0$
2 **si** $x = 0$ **alors** aller à l'instruction 6
3 $x \leftarrow x - 1$
4 $z \leftarrow z + 1$
5 aller à l'instruction 2
6 **si** $y = 0$ **alors** aller à l'instruction 10
7 $y \leftarrow y - 1$
8 $z \leftarrow z + 2$
9 aller à l'instruction 6
10 $z \leftarrow z + 0$

générale, les machines qui possèdent le test à zéro donnent donc lieu à des problèmes indécidables. Bien qu'il existe des modèles de machines à compteurs où certains tests à zéro ne mènent pas à la pleine puissance des machines de Turing [FS00, HKOW09], nous ne les étudierons pas.

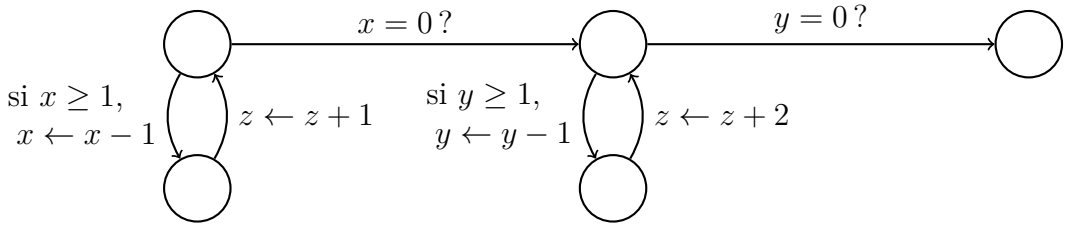


Figure 2.3 – Exemple de machine à compteur, sous forme d'automate, calculant $x + 2y$.

Dans cette thèse, nous étudierons principalement les systèmes d'addition de vecteurs et les réseaux de Petri. Ces deux modèles de machines à compteurs, dont nous verrons l'équivalence à la section 2.9.3, sont munis de l'addition et de la soustraction (conditionnelle ou non selon la sémantique). Ces deux modèles sont capturés par la classe plus générale des réseaux affines. Nous introduisons brièvement cette classe puisqu'elle sera mentionnée au chapitre 3 et elle nous permet de montrer en une seule preuve que toutes ses sous-classes sont bien structurées.

Definition 2.19 ([FMP04])**.** Soit $d \in \mathbb{N}_{>0}$. Un *réseau affine à d compteurs (avec états)* est une paire (Q, T) telle que

- Q est un ensemble fini d'éléments (*états de contrôle*),
- $T \subseteq Q \times (\mathbb{N}^{d\times d} \times \mathbb{Z}^d) \times Q$ est un ensemble fini (*transitions*).

Soit $t = (p, (\mathbf{A}, \boldsymbol{b}), q) \in T$, posons $f_t : Q \times \mathbb{N}^d \to Q \times \mathbb{N}^d$ la fonction partielle telle que

$$f_t(r, \boldsymbol{x}) \stackrel{\text{déf}}{=} \begin{cases} (s, \mathbf{A}\boldsymbol{x} + \boldsymbol{b}) & \text{si } r = p,\ s = q, \text{ et } \mathbf{A}\boldsymbol{x} + \boldsymbol{b} \geq \mathbf{0}, \\ \text{non définie} & \text{sinon.} \end{cases}$$

Le réseau $\mathcal{V}$ induit le système de transitions fonctionnel et ordonné $\mathcal{S} = (Q \times \mathbb{N}^d, \to, F, \leq)$ où $F \stackrel{\text{déf}}{=} \{f_t : t \in T\}$.

Proposition 2.20. *Soit $d \in \mathbb{N}_{>0}$. L'ensemble des réseaux affines à d compteurs forme une classe post-effective de WSTS à branchement fini avec monotonicité forte. De plus, toute sous-classe qui respecte la propriété suivante possède additionnellement la monotonicité stricte : pour chaque matrice $\mathbf{A}$, $\boldsymbol{x} > \mathbf{0}$ implique $\mathbf{A}\boldsymbol{x} > \mathbf{0}$.*

Démonstration. Soit $\mathcal{C}$ l'ensemble des réseaux affines à d compteurs. Nous supposons, sans perte de généralité, qu'il existe une machine de Turing qui sur entrée $i, j \in \mathbb{N}$ donne la description finie de la $j^{\text{ème}}$ fonction partielle de $\mathcal{C}(i)$.

Soit $\mathcal{S} = (Q \times \mathbb{N}^d, \to, F, \leq) \in \mathcal{C}$, montrons d'abord que $\mathcal{S}$ est un WSTS avec monotonicité forte. Puisque $\leq$ est un beau préordre pour $\mathbb{N}^d$ et que $\mathcal{S}$ est fonctionnel, par la proposition 2.17, il suffit de montrer que F est un ensemble de fonctions partielles non décroissantes (ou croissantes si l'on désire obtenir la monotonicité stricte). Soient $f \in F$, $p \in Q$, et $\boldsymbol{x} \in \mathbb{N}^d$ tels que $f(p, \boldsymbol{x})$ est définie. Nous avons $f(p, \boldsymbol{x}) = (q, \mathbf{A}\boldsymbol{x} + \boldsymbol{b})$ pour certains $q \in Q, \mathbf{A} \in \mathbb{N}^{d\times d}$ et $\boldsymbol{b} \in \mathbb{Z}^d$. Soit $\boldsymbol{y} > \boldsymbol{x}$. Nous

avons,

$$\begin{aligned} f(p, \boldsymbol{y}) &= (q, \mathbf{A}\boldsymbol{y} + \boldsymbol{b}) \\ &= (q, \mathbf{A}(\boldsymbol{x} + \boldsymbol{z}) + \boldsymbol{b}) && \text{(pour un certain } \boldsymbol{z} > \mathbf{0}) \\ &= (q, \mathbf{A}\boldsymbol{x} + \mathbf{A}\boldsymbol{z} + \boldsymbol{b}) \\ &\geq (q, \mathbf{A}\boldsymbol{x} + \boldsymbol{b}) && \text{(car } \mathbf{A}\boldsymbol{z} \geq \mathbf{0}) \\ &= f(p, \boldsymbol{x}) \ . \end{aligned}$$

Notons que dans l'argument précédent, si $\mathbf{A}\boldsymbol{z} > \mathbf{0}$, alors $f(p, \boldsymbol{y}) > f(p, \boldsymbol{x})$.

Nous remarquons que $|\mathsf{Succ}_{\mathcal{S}}(p, \boldsymbol{x})| \leq |F|$ pour tous $p \in Q$ et $\boldsymbol{x} \in \mathbb{N}^d$, ainsi $\mathcal{S}$ est à branchement fini. Nous remarquons également que $\mathcal{C}$ est une classe post-effective. En effet, l'ordre $\leq$ est le même pour tout $\mathcal{S} \in \mathcal{C}$ et peut être facilement calculé en temps polynomial avec l'arithmétique en binaire. De plus, le calcul des relations de transition $\xrightarrow{f}$ ne dépend essentiellement que d'un produit matriciel, d'une addition de vecteurs et d'un test de non négativité ; toutes ces opérations peuvent être calculées en temps polynomial. □

2.9.1 Systèmes d'addition de vecteurs

Definition 2.21. Soit $d \in \mathbb{N}_{>0}$. Un *système d'addition de vecteurs (avec états[9]) à d compteurs*, abrégé *d-VASS*, est une paire $\mathcal{V} = (Q, T)$ telle que

- Q est un ensemble fini (*états de contrôle*), et
- $T \subseteq Q \times \mathbb{Z}^d \times Q$ est un ensemble fini (*transitions*).

Lorsque le nombre de compteurs d n'est pas pertinent, nous disons simplement que $\mathcal{V}$ est un système d'addition de vecteurs.

[9]Dans cette thèse, nous ne distinguerons pas les termes « système d'addition de vecteurs avec états » et « systèmes d'addition de vecteurs » utilisés dans la littérature ; les « systèmes d'addition de vecteurs » constitués seulement d'un ensemble de vecteurs dans la littérature peuvent être simplement vus ici comme un cas particulier où $|Q| = 1$.

Un système d'addition de vecteurs peut être illustré graphiquement par un graphe dirigé où les états de contrôle forment les sommets. Un arc de p vers q est ajouté si $(p, \boldsymbol{z}, q) \in T$ et est étiqueté par $\boldsymbol{z}$. Par exemple, la figure 2.4 représente le système d'addition de vecteurs $\mathcal{V} = (Q, T)$ tel que $Q = \{p, q\}$ et

$$\begin{aligned} T = \{&(p,\ (\ \ 2,\ \ 2),\ p),\\ &(p,\ (-5, -5),\ q),\\ &(q,\ (\ \ 0,\ \ 1),\ q),\\ &(q,\ (-2,\ \ 0),\ p)\}\ . \end{aligned}$$

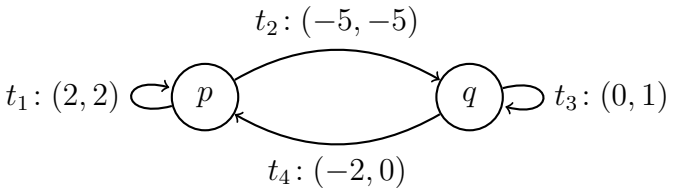


Figure 2.4 – Exemple de système d'addition de vecteurs à deux compteurs.

De façon générale, nous associerons à un système d'addition de vecteurs à d compteurs $\mathcal{V} = (Q, T)$ le système de transitions ordonné $\mathcal{S} = (Q \times \mathbb{Z}^d, \rightarrow, T, \leq)$ tel que $p(\boldsymbol{u}) \xrightarrow{t} q(\boldsymbol{v})$ lorsque $t = (p, \boldsymbol{z}, q) \in T$ et $\boldsymbol{u} + \boldsymbol{z} = \boldsymbol{v}$. Le système de transitions le plus intéressant, et celui normalement considéré dans la littérature, est celui obtenu sur $Q \times \mathbb{N}^d$. Toutefois, dans cette thèse, afin de raisonner sur $\mathbb{N}^d$, nous devrons d'abord raisonner sur $\mathbb{Z}^d$. Nous ferons ainsi une distinction entre la *sémantique standard*, c.-à-d. celle où l'ensemble d'états est $Q \times \mathbb{N}^d$, et la *sémantique entière*, c.-à-d. celle où l'ensemble d'états est $Q \times \mathbb{Z}^d$. Notons que dans le cas de la sémantique standard, une transition ne peut être empruntée que si le vecteur résultant est non négatif.

En général, nous supposerons implicitement que nous travaillons sous la sémantique entière et nous ferons plutôt usage de la notation $\rightarrow_{\mathbb{A}}$ introduite à la section 2.8.1 avec, entre autres, $\mathbb{A} = Q \times \mathbb{Z}^d$ et $\mathbb{A} = Q \times \mathbb{N}^d$, et par abus de notation, nous écrirons $\mathbb{A} = X$ où $X \subseteq \mathbb{Z}^d$ pour $\mathbb{A} = Q \times X$. Reconsidérons le système

d'addition de vecteurs de la figure 2.4. Nous avons

$$p(3,2) \xrightarrow{t_1}_{\mathbb{N}^2} p(5,4) \xrightarrow{t_1}_{\mathbb{N}^2} p(7,6) \xrightarrow{t_2}_{\mathbb{N}^2} q(2,1) \xrightarrow{t_3}_{\mathbb{N}^2} q(2,2) \xrightarrow{t_4}_{\mathbb{N}^2} p(0,2) \ ,$$

et ainsi $\pi = t_1t_1t_2t_3t_4$ est un chemin restreint à $\mathbb{N}^2$ de $p(3,2)$ vers $p(0,2)$. Notons que $\pi' = t_2t_3t_4t_1t_1$, qui est une permutation de π, est un chemin à partir de $p(3,2)$ dans $\mathbb{Z}^2$; cependant cela n'est pas le cas de π' lorsque restreint à $\mathbb{N}^2$.

Sous la sémantique entière, les d-VASS ne sont pas des systèmes de transitions bien structurés puisque $\leq$ n'est pas un beau préodre pour $\mathbb{Z}^d$. Cependant, il est simple de voir qu'un d-VASS $\mathcal{V} = (Q, T)$ sous sa sémantique standard est équivalent au réseau affine à d compteurs (Q, T') tel que $T' = \{(p, (\mathbf{I}, \boldsymbol{z}), q) : (p, \boldsymbol{z}, q) \in T\}$ où $\mathbf{I}$ est la matrice identité. Ainsi, par la proposition 2.20, l'ensemble des d-VASS interprétés sous leur sémantique standard forme une classe post-effective de WSTS à branchement fini avec monotonicité forte et stricte.

2.9.2 Réseaux de Petri

Definition 2.22. Un *réseau de Petri* est un tuplet $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ tel que

- P est un ensemble fini (*places*),
- T est un ensemble fini tel que $P \cap T = \emptyset$ (*transitions*),
- $\mathbf{Pre} \in \mathbb{N}^{P \times T}$ (*matrice d'incidence arrière*),
- $\mathbf{Post} \in \mathbb{N}^{P \times T}$ (*matrice d'incidence avant*).

Un réseau de Petri peut être illustré graphiquement par un graphe dirigé où les places et les transitions forment les sommets. Un arc de p vers t est ajouté si $\mathbf{Pre}(p, t) > 0$ et est étiqueté par $\mathbf{Pre}(p, t)$. Similairement, un arc de t vers p est ajouté si $\mathbf{Post}(p, t) > 0$ et est étiqueté par $\mathbf{Post}(p, t)$. Par exemple, la figure 2.5 représente le réseau de Petri $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ tel que $P = \{p, q, r\}$, $T =$

$\{s, t\}$,

$$\mathbf{Pre} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \qquad \text{et} \qquad \mathbf{Post} = \begin{pmatrix} 0 & 2 \\ 0 & 0 \\ 2 & 1 \end{pmatrix} .$$

Par convention, nous n'étiquetons par les arcs dont l'étiquette devrait prendre la valeur 1.

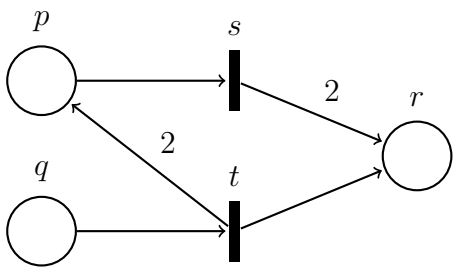


Figure 2.5 – Exemple de réseau de Petri.

Soit $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri. Un *marquage discret*, ou simplement *marquage*, de $\mathcal{N}$ est un vecteur $\boldsymbol{m} \in \mathbb{N}^P$. Pour un marquage $\boldsymbol{m}$ et une place p, nous disons que p possède $\boldsymbol{m}(p)$ *jetons*. Graphiquement, nous illustrerons les jetons par des cercles ($\bullet$). La *matrice d'incidence* **Incid** de $\mathcal{N}$ est la matrice entière définie par

$$\mathbf{Incid} \stackrel{\text{déf}}{=} \mathbf{Post} - \mathbf{Pre}.$$

L'*inverse* de $\mathcal{N}$, dénoté $\mathcal{N}^{-1}$, est obtenu en inversant **Pre** et **Post**, c.-à-d.

$$\mathcal{N}^{-1} \stackrel{\text{déf}}{=} (P, T, \mathbf{Post}, \mathbf{Pre}).$$

Soit $t \in T$, les *places d'entrée*, les *places de sortie* et les *voisins* de t sont respecti-

vement définis par

$$\begin{aligned} {}^\bullet t &\stackrel{\text{déf}}{=} \{p \in P : \ \mathbf{Pre}(p,t) > 0\} \ , \\ t^\bullet &\stackrel{\text{déf}}{=} \{p \in P : \mathbf{Post}(p,t) > 0\} \ , \text{ et} \\ {}^\bullet t^\bullet &\stackrel{\text{déf}}{=} {}^\bullet t \cup t^\bullet \ . \end{aligned}$$

Soit $p \in P$, de façon similaire, les *transitions d'entrée*, les *transitions de sortie* et les *voisins* de p sont respectivement définis par

$$\begin{aligned} {}^\bullet p &\stackrel{\text{déf}}{=} \{t \in T : \mathbf{Post}(p,t) > 0\} \ , \\ p^\bullet &\stackrel{\text{déf}}{=} \{t \in T : \ \mathbf{Pre}(p,t) > 0\} \ , \text{ et} \\ {}^\bullet p^\bullet &\stackrel{\text{déf}}{=} {}^\bullet p \cup p^\bullet \ . \end{aligned}$$

Cette notation est étendue naturellement aux ensembles de transitions et de places, p. ex., pour $T' \subseteq T$ nous avons ${}^\bullet T' = \bigcup_{t \in T'} {}^\bullet t$. Considérons à nouveau le réseau de Petri illustré à la figure 2.5, nous avons, par exemple,

$$\begin{aligned} {}^\bullet p &= \emptyset \ , & q^\bullet &= \{t\} \ , \\ {}^\bullet s &= \{p\} \ , & t^\bullet &= \{r\} \ , \\ {}^\bullet r &= \{s,t\} \ , \text{et} & {}^\bullet t^\bullet &= \{q,r\} \ . \end{aligned}$$

Soit $T' \subseteq T$, le (sous-)réseau de Petri $\mathcal{N}_{T'}$ induit par T' est

$$\mathcal{N}_{T'} \stackrel{\text{déf}}{=} (P', T', \mathbf{Pre}_{P' \times T'}, \mathbf{Post}_{P' \times T'})$$

où $P' = {}^\bullet T'^\bullet$.

Nous disons que $t \in T$ est *franchissable* en un marquage $\boldsymbol{m}$ lorsque $\boldsymbol{m}(p) \geq \mathbf{Pre}(p,t)$ pour tout $p \in P$. Une telle transition franchissable peut être *activée*,

auquel cas elle mène au nouveau marquage $\boldsymbol{m}'$ tel que, pour tout $p \in P$,

$$\begin{aligned}\boldsymbol{m}'(p) &= \boldsymbol{m} - \mathbf{Pre}(p,t) + \mathbf{Post}(p,t) \\ &= \boldsymbol{m} + \mathbf{Incid}(p,t)\ .\end{aligned}$$

Nous associons à $\mathcal{N}$ le système de transitions ordonné $\mathcal{S} = (\mathbb{N}^d, \to, T, \leq)$ tel que $\boldsymbol{m} \xrightarrow{t} \boldsymbol{m}'$ lorsque t est franchissable en $\boldsymbol{m}$, et $\boldsymbol{m}'$ est le marquage obtenu lors de l'activation de t en m.

Reconsidérons le réseau de Petri de la figure 2.5. Nous avons $(0,1,0) \xrightarrow{t} (0,2,1) \xrightarrow{s} (0,1,3) \xrightarrow{s} (0,0,5)$, et ainsi tss est un chemin de $(0,1,0)$ vers $(0,0,5)$. Notons que sts n'est pas un chemin à partir de $(0,1,0)$; en effet, il est impossible d'activer d'abord la transition s puisque la place p ne contient initialement aucun jeton. Les transitions ne commutent ainsi généralement pas dans un réseau de Petri.

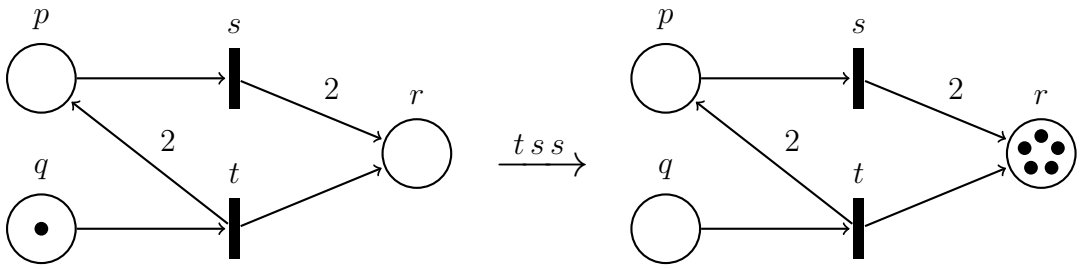


Figure 2.6 – Exemple d'activation des transitions t, s et s du réseau de Petri de la figure 2.5 à partir du marquage $(0,1,0)$.

Tout réseau de Petri à d places possède un réseau affine à d compteurs équivalent. Ils forment ainsi une classe post-effective de WSTS à branchement fini avec monotonicité forte et stricte. Nous formaliserons cette observation à la section 2.9.3 en montrant que les réseaux de Petri sont en fait équivalents aux systèmes d'addition de vecteurs.

2.9.3 Équivalences entre modèles

Dans cette section, nous expliquons comment passer d'un système d'addition de vecteurs à un réseau de Petri, et vice-versa, de façon à préserver les différents comportements. Bien que les constructions présentées soient bien connues (voir par ex. [Reu88, chapitre 1]), nous prenons soin de détailler leurs propriétés afin de démontrer l'équivalence, au sens calculatoire, des différents problèmes de décision des systèmes d'addition de vecteurs et des réseaux de Petri.

Comme étape intermédiaire, nous rappelons d'abord comment passer d'un VASS à un VASS à un seul état. Une construction simple permet de transformer un d-VASS $\mathcal{V} = (Q, T)$ en un $(d+2|Q|)$-VASS qui possède un seul état de contrôle et qui « encode » Q dans ses $2|Q|$ compteurs additionnels [Reu88, p. 18]. Il est cependant possible de faire mieux. Tel que remarqué par Hopcroft & Pansiot [HP79], Q peut être représenté par trois nouveaux compteurs :

Lemme 2.23 ([HP79])**.** *Soit $\mathcal{V} = (Q, T)$ un d-VASS. Posons $B = |Q|(|Q|+1)]$. Il existe un $(d+3)$-VASS $\mathcal{V}' = (\{r\}, T')$ à un seul état de contrôle, et un ensemble de vecteurs $\{\boldsymbol{a}_p : p \in Q\} \subseteq [0, B]^3$, calculables en espace logarithmique à partir de $\mathcal{V}$, tels que*

(a) $|T'| = 2|Q| + |T|$,

(b) $T' \subseteq \{r\} \times ([-\|T\|, \|T\|]^d \times [-B, B]^3) \times \{r\}$,

(c) $p(\boldsymbol{u}) \to^k_{\mathbb{N}^d} q(\boldsymbol{v})$ *dans* $\mathcal{V}$ *si, et seulement si,* $r(\boldsymbol{u}\boldsymbol{a}_p) \to^{3k}_{\mathbb{N}^{d'}} r(\boldsymbol{v}\boldsymbol{a}_q)$ *dans* $\mathcal{V}'$*, et*

(d) si $r(\boldsymbol{u}\boldsymbol{a}_p) \to^k_{\mathbb{N}^{d'}} r(\boldsymbol{v}\boldsymbol{a}_q)$ *dans* $\mathcal{V}'$*, alors* $k \bmod 3 = 0$*, et*

(e) si $r(\boldsymbol{u}\boldsymbol{a}_p) \to^k_{\mathbb{N}^{d'}} r(\boldsymbol{y})$ *dans* $\mathcal{V}'$*, alors il existe* $\boldsymbol{v} \in \mathbb{N}^d$ *et* $q \in Q$ *tels que* $r(\boldsymbol{u}\boldsymbol{a}_p) \to^{k'}_{\mathbb{N}^{d'}} r(\boldsymbol{v}\boldsymbol{a}_q)$ *dans* $\mathcal{V}'$ *où* $k' = 3(k \div 3)$.

Les deux propositions suivantes montrent comment passer d'un VASS à un réseau de Petri et vice-versa.

Proposition 2.24. *Soit $\mathcal{V} = (Q, T)$ un d-VASS. Posons $B = |Q|(|Q| + 1)]$. Il existe un réseau de Petri $\mathcal{N} = (P, S, \mathbf{Pre}, \mathbf{Post})$ et un ensemble de vecteurs $\{\boldsymbol{a}_p : p \in Q\} \subseteq [0, B]^3$, calculables en espace logarithmique à partir de $\mathcal{V}$, tels que*

(a) $|P| = d + 3$,

(b) $|S| = 2|Q| + |T|$,

(c) $\mathbf{Pre}[P \times \{s\}], \mathbf{Post}[P \times \{s\}] \in [0, \|T\|]^d \times [0, B]^3$ *pour tout* $s \in S$,

(d) $p(\boldsymbol{u}) \to^k_{\mathbb{N}^d} q(\boldsymbol{v})$ *dans* $\mathcal{V}$ *si, et seulement si,* $\boldsymbol{u}\boldsymbol{a}_p \to^{3k} \boldsymbol{v}\boldsymbol{a}_q$ *dans* $\mathcal{N}$, *et*

(e) si $\boldsymbol{u}\boldsymbol{a}_p \to^k \boldsymbol{v}\boldsymbol{a}_q$ *dans* $\mathcal{N}$, *alors* $k \bmod 3 = 0$, *et*

(f) si $\boldsymbol{u}\boldsymbol{a}_p \to^k \boldsymbol{y}$ *dans* $\mathcal{N}$, *alors il existe* $\boldsymbol{v} \in \mathbb{N}^d$ *et* $q \in Q$ *tels que* $\boldsymbol{u}\boldsymbol{a}_p \to^{k'} \boldsymbol{v}\boldsymbol{a}_q$ *dans* $\mathcal{N}$ *où* $k' = 3(k \div 3)$.

Démonstration. Soit $\mathcal{V} = (Q, T)$ un d-VASS. Soient $\mathcal{V}' = (\{r\}, T')$ et $\{\boldsymbol{a}_p : p \in Q\} \subseteq \mathbb{N}^3$ respectivement le VASS à $d+3$ compteurs et l'ensemble obtenus à partir de $\mathcal{V}$ au lemme 2.23. Nous avons $T' = \{t_1, t_2, \ldots, t_n\}$ où $1 \leq n \leq 2|Q| + |T|$. Nous utilisons une transformation standard afin de transformer un VASS à un seul état de contrôle en un réseau de Petri. Cette transformation est illustrée dans le cas $n = 3$ à la figure 2.7. Plus formellement, soient $i \in [n]$ et $t_i = (p, \boldsymbol{z}_i, q)$, nous posons $\boldsymbol{z}_i^-, \boldsymbol{z}_i^+ \in \mathbb{N}^d$ les vecteurs tels que

$$\begin{aligned} \boldsymbol{z}_i^-(j) &\stackrel{\text{déf}}{=} |\min(\boldsymbol{z}_i(j), 0)|, \\ \boldsymbol{z}_i^+(j) &\stackrel{\text{déf}}{=} \max(\boldsymbol{z}_i(j), 0)\ . \end{aligned}$$

Autrement dit, nous divisons le vecteur $\boldsymbol{z}_i$ en ses portions négative et positive de telle sorte que $\boldsymbol{z}_i = -\boldsymbol{z}_i^- + \boldsymbol{z}_i^+$.

Nous posons $\mathcal{N} = (P, S, \mathbf{Pre}, \mathbf{Post})$ où $P \stackrel{\text{déf}}{=} \{p_i : i \in [d+3]\}$, $S \stackrel{\text{déf}}{=} \{s_i : i \in [n]\}$,

et pour tout $i \in [n]$,

$$\mathbf{Pre}[P \times \{s_i\}] \stackrel{\text{déf}}{=} \boldsymbol{z}_i^-, \text{ et}$$
$$\mathbf{Post}[P \times \{s_i\}] \stackrel{\text{déf}}{=} \boldsymbol{z}_i^+.$$

Nous avons $r(\boldsymbol{x}) \xrightarrow{t_i}_{\mathbb{N}^{d+3}} r(\boldsymbol{y})$ dans $\mathcal{V}'$ si, et seulement si, $\boldsymbol{x} \xrightarrow{s_i} \boldsymbol{y}$ dans $\mathcal{N}$. Ainsi, par le lemme 2.23 et par induction sur k, les points (d), (e) et (f) sont satisfaits. □

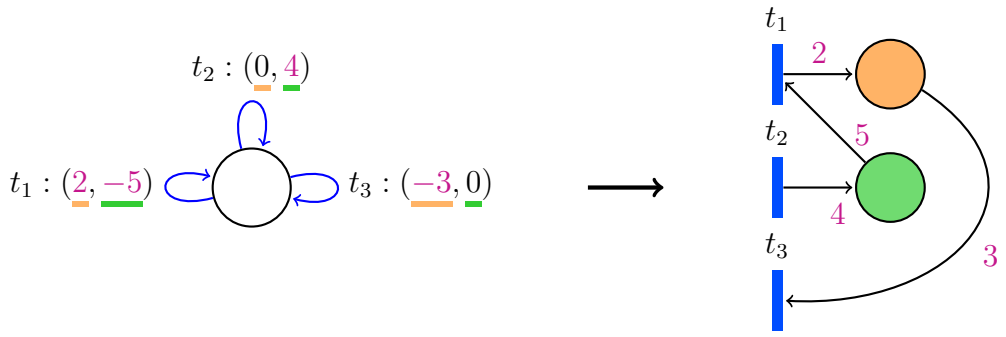


Figure 2.7 – Exemple de transformation d'un VASS à un seul état de contrôle en un réseau de Petri.

Proposition 2.25. *Soit $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri. Il existe un d-VASS $\mathcal{V} = (Q, T')$, calculable en espace logarithmique à partir de $\mathcal{N}$, tel que*

(a) $d = |P|$,

(b) $Q = \{c\} \cup \{p_t : t \in T\}$,

(c) $|T'| \leq 2|T|$,

(d) $\boldsymbol{u} \rightarrow^k \boldsymbol{v}$ *dans* $\mathcal{N}$ *si, et seulement si,* $c(\boldsymbol{u}) \rightarrow^{2k}_{\mathbb{N}^d} c(\boldsymbol{v})$ *dans* $\mathcal{V}$, *et*

(e) si $c(\boldsymbol{u}) \rightarrow^k_{\mathbb{N}^d} c(\boldsymbol{v})$ *dans* $\mathcal{V}$, *alors* $k \bmod 2 = 0$,

Démonstration. Soit $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri. Nous pouvons supposer, sans perte de généralité, que $P = [d]$ où $d \stackrel{\text{déf}}{=} |P|$. Nous utilisons une transformation standard afin de transformer un réseau de Petri en un VASS. Cette

transformation est illustrée à la figure 2.8. Formellement, nous posons $\mathcal{V} = (Q, T')$ le d-VASS tel que $Q \stackrel{\text{déf}}{=} \{c\} \cup \{p_t : t \in T\}$ et

$$T' \stackrel{\text{déf}}{=} \{(c, -\mathbf{Pre}[P \times \{t\}], p_t) : t \in T\} \cup \{(p_t, \mathbf{Post}[P \times \{t\}], c) : t \in T\} .$$

Soient $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{N}^d$. Supposons que $\boldsymbol{u} \xrightarrow{t} \boldsymbol{v}$ dans $\mathcal{N}$, alors, par définition de T', $c(\boldsymbol{u}) \to_{\mathbb{N}^d} p_t(\boldsymbol{w}) \to_{\mathbb{N}^d} c(\boldsymbol{v})$ dans $\mathcal{V}$ où $\boldsymbol{w} = \boldsymbol{u} - \mathbf{Pre}[P \times \{t\}]$. Supposons que $c(\boldsymbol{u}) \to^2_{\mathbb{N}^d} c(\boldsymbol{v})$, alors forcément $c(\boldsymbol{u}) \to_{\mathbb{N}^d} p_t(\boldsymbol{w}) \to_{\mathbb{N}^d} c(\boldsymbol{v})$ pour un certain $t \in T$ et $\boldsymbol{w} \in \mathbb{N}^d$. De plus, par définition de $\mathcal{V}$, nous avons $\boldsymbol{w} = \boldsymbol{u} - \mathbf{Pre}[P \times \{t\}]$. Ainsi, $\boldsymbol{u} \xrightarrow{t} \boldsymbol{v}$ dans $\mathcal{N}$. Ainsi, par induction sur k, le point (d) est satisfait. Notons que le point (e) est satisfait par définition de $\mathcal{V}$. □

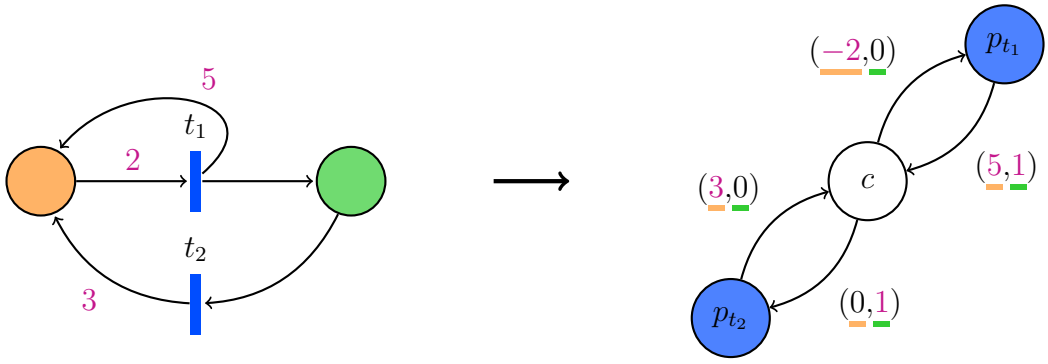


Figure 2.8 – Exemple de transformation d'un réseau de Petri en un VASS.

Nous sommes maintenant en mesure de prouver formellement ce que nous entendons par équivalence entre les VASS et les réseaux de Petri.

Proposition 2.26. *Le problème d'accessibilité (resp. couverture, terminaison, finitude) pour les VASS, sous la sémantique standard, est équivalent au problème d'accessibilité (resp. couverture, terminaison, finitude) pour les réseaux de Petri.*

Démonstration. Nous quelques réductions parmi les huit réductions afin d'illustrer les différents détails techniques.

$\underline{\text{Accessibilité}_{\text{VASS}} \leq_m^{\log} \text{Accessibilité}_{\text{Petri}}}$. Soient $\mathcal{V} = (Q, T)$ un d-VASS et $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^d$. Soient $\mathcal{N} = (P, T', \mathbf{Pre}, \mathbf{Post})$ le réseau de Petri et $\{\boldsymbol{a}_p : p \in Q\}$ l'ensemble de vecteurs, calculables en espace logarithmique, obtenus à la proposition 2.24. Nous montrons que $p(\boldsymbol{u}) \to^*_{\mathbb{N}^d} q(\boldsymbol{v})$ dans $\mathcal{V}$ si, et seulement si, $\boldsymbol{u}\boldsymbol{a}_p \to^* \boldsymbol{v}\boldsymbol{a}_q$ dans $\mathcal{N}$.

$\Rightarrow$) Supposons que $p(\boldsymbol{u}) \to^*_{\mathbb{N}^d} q(\boldsymbol{v})$ dans $\mathcal{V}$, alors par le point (d) de la proposition 2.24, $\boldsymbol{u}\boldsymbol{a}_p \to^* \boldsymbol{v}\boldsymbol{a}_q$ dans $\mathcal{N}$.
$\Leftarrow$) Supposons que $\boldsymbol{u}\boldsymbol{a}_p \to^* \boldsymbol{v}\boldsymbol{a}_q$ dans $\mathcal{N}$, alors par le point (e) de la proposition 2.24, $\boldsymbol{u}\boldsymbol{a}_p \to^{3m} \boldsymbol{v}\boldsymbol{a}_q$ dans $\mathcal{N}$ pour un certain m, et ainsi par le point (d), $p(\boldsymbol{u}) \to^*_{\mathbb{N}^d} q(\boldsymbol{v})$ dans $\mathcal{V}$.

$\underline{\text{Terminaison}_{\text{Petri}} \leq_m^{\log} \text{Terminaison}_{\text{VASS}}}$. Soit $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri et $\boldsymbol{u} \in \mathbb{N}^d$ où $d = |P|$. Soient $\mathcal{V} = (\{c\} \cup \{p_t : t \in T\}, T')$ le d-VASS, calculable en espace logarithmique, obtenu à la proposition 2.25. Montrons que $\mathsf{Exec}^{\mathsf{inf}}_{\mathcal{N}}(\boldsymbol{u}) = \emptyset \iff \mathsf{Exec}^{\mathsf{inf}}_{\mathcal{V}}(c(\boldsymbol{u})) = \emptyset$.

$\Rightarrow$) Supposons que $\mathsf{Exec}^{\mathsf{inf}}_{\mathcal{N}}(\boldsymbol{u}) = \emptyset$, alors il existe une borne $m \in \mathbb{N}$ sur la longueur des exécutions, et donc des chemins, à partir de $\boldsymbol{u}$ dans $\mathcal{N}$. Afin d'obtenir une contradiction, supposons que $\mathsf{Exec}^{\mathsf{inf}}_{\mathcal{V}}(c(\boldsymbol{u})) \neq \emptyset$. Il existe donc $\boldsymbol{v} \in \mathbb{N}^d$ et $k > 2m$ tels que $c(\boldsymbol{u}) \to^k_{\mathbb{N}^d} c(\boldsymbol{v})$ dans $\mathcal{V}$. Par le point (e) de la proposition 2.25, $k = 2k'$ pour un certain $k' \in \mathbb{N}$. Par le point (d) de la proposition 2.25, $\boldsymbol{u} \to^{k'} \boldsymbol{v}$ dans $\mathcal{N}$. Or, $2k' = k > 2m$ et ainsi $k' > m$, ce qui est une contradiction.

$\Leftarrow$) Supposons que $\mathsf{Exec}^{\mathsf{inf}}_{\mathcal{V}}(c(\boldsymbol{u})) = \emptyset$, alors il existe une borne $m \in \mathbb{N}$ sur la longueur des exécutions, et donc des chemins, à partir de $c(\boldsymbol{u})$ dans $\mathcal{V}$. Afin d'obtenir une contradiction, supposons que $\mathsf{Exec}^{\mathsf{inf}}_{\mathcal{N}}(\boldsymbol{u}) \neq \emptyset$. Il existe donc $\boldsymbol{v} \in \mathbb{N}^d$ et $k > m$ tels que $\boldsymbol{u} \to^k \boldsymbol{v}$ dans $\mathcal{N}$. Par le point (d) de la proposition 2.25, $c(\boldsymbol{u}) \to^{2k}_{\mathbb{N}^d} c(\boldsymbol{v})$ dans $\mathcal{V}$, ce qui est une contradiction.

$\underline{\text{Finitude}_{\text{VASS}} \leq_m^{\log} \text{Finitude}_{\text{Petri}}}$. Soient $\mathcal{V} = (Q, T)$ un d-VASS et $p(\boldsymbol{u}) \in Q \times \mathbb{N}^d$. Soient $\mathcal{N} = (P, T', \mathbf{Pre}, \mathbf{Post})$ le réseau de Petri et $\{\boldsymbol{a}_p : p \in Q\}$ l'ensemble de vecteurs, calculables en espace logarithmique, obtenus à la proposition 2.24.

Montrons que $\mathsf{Succ}^*_{\mathcal{V}}(p(\boldsymbol{u}))$ est fini si, et seulement si, $\mathsf{Succ}^*_{\mathcal{N}}(\boldsymbol{u}\boldsymbol{a}_p)$ est fini.

$\Rightarrow$) Supposons que $\mathsf{Succ}^*_{\mathcal{V}}(p(\boldsymbol{u}))$ soit fini. Afin d'obtenir une contradiction, supposons que $\mathsf{Succ}^*_{\mathcal{N}}(\boldsymbol{u}\boldsymbol{a}_p)$ soit infini. Il existe donc une suite infinie $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$ de marquages distincts, et une suite infinie d'indices $k_0 < k_1 < \ldots \in \mathbb{N}$ tels que, pour tout $i \in \mathbb{N}$, $\boldsymbol{u}\boldsymbol{a}_p \to^{k_i} \boldsymbol{x}_i$ dans $\mathcal{N}$. Nous pouvons supposer, sans perte de généralité, qu'il existe $r \in \{0, 1, 2\}$ tel que pour tout $i \in \mathbb{N}$, $\boldsymbol{x}_i \bmod 3 = r$. Par le point (f) de la proposition 2.24, pour tout $i \in \mathbb{N}$, il existe $\boldsymbol{v}_i \in \mathbb{N}^d$ et $q_i \in Q$ tels que $\boldsymbol{u}\boldsymbol{a}_p \to^{k'_i} \boldsymbol{v}_i\boldsymbol{a}_{q_i}$ dans $\mathcal{N}$, où $k'_i = 3(k_i \div 3)$. En particulier, nous avons donc $\boldsymbol{u}\boldsymbol{a}_p \to^{k'_i} \boldsymbol{v}_i\boldsymbol{a}_{q_i} \to^{r} \boldsymbol{x}_i$. Puisque $\mathcal{N}$ ne possède qu'un nombre fini de transitions et que $\{\boldsymbol{x}_i : i \in \mathbb{N}\}$ est infini, l'ensemble $\{\boldsymbol{v}_i\boldsymbol{a}_{q_i} : i \in \mathbb{N}\}$ est donc forcément infini. Par le point (d) de la proposition 2.24, nous avons $p(\boldsymbol{u}) \to^{k_i \div 3} q_i(\boldsymbol{v}_i)$ dans $\mathcal{V}$. Ainsi $\{q_i(\boldsymbol{v}_i) : i \in \mathbb{N}\}$ est un sous-ensemble infini de $\mathsf{Succ}^*_{\mathcal{V}}(p(\boldsymbol{u}))$, ce qui est une contradiction.

$\Leftarrow$) Supposons que $\mathsf{Succ}^*_{\mathcal{N}}(\boldsymbol{u}\boldsymbol{a}_p)$ soit fini. Afin d'obtenir une contradiction, supposons que $\mathsf{Succ}^*_{\mathcal{V}}(p(\boldsymbol{u}))$ soit infini. Il existe donc une suite infinie $q_0(\boldsymbol{v}_0), q_1(\boldsymbol{v}_1), \ldots$ d'éléments distincts tels que, pour tout $i \in \mathbb{N}$, $p(\boldsymbol{u}) \to^* q_i(\boldsymbol{v}_i)$ dans $\mathcal{V}$. Par le point (d) de la proposition 2.24, pour tout $i \in \mathbb{N}$, $\boldsymbol{u}\boldsymbol{a}_p \to^* \boldsymbol{v}_i\boldsymbol{a}_{q_i}$ dans $\mathcal{N}$. Ainsi, $\{\boldsymbol{v}_i\boldsymbol{a}_{q_i} : i \in \mathbb{N}\}$ est un sous-ensemble infini de $\mathsf{Succ}^*_{\mathcal{N}}(\boldsymbol{u}\boldsymbol{a}_p)$, ce qui est une contradiction. □

3

DÉCIDABILITÉ ET INDÉCIDABILITÉ DANS LES WSTS À BRANCHEMENT INFINI

Dans ce chapitre, nous étudions la complexité des problèmes de décision (voir section 2.8.5) pour les systèmes de transitions bien structurés. L'essentiel des techniques et des résultats de décidabilité que l'on retrouve dans la littérature ne s'appliquent qu'aux WSTS à branchement fini. Or, il existe plusieurs modèles de WSTS à branchement infini : les automates à insertion [CFI96], les automates à insertion avec files [BMO+12], les systèmes parallèles récursifs [KPS96], les réseaux de Petri-ω [GHPR15], etc. Mentionnons également les WSTS paramétrés, c.-à-d. les WSTS munis de variables qui peuvent prendre une valeur tirée d'un ensemble infini. Ces modèles de WSTS s'avèrent souvent utiles pour l'analyse de systèmes. Par exemple, Geeraerts *et al.* montrent dans [GHPR15] comment les systèmes concurrents paramétrés, pouvant créer des fils d'execution dynamiquement, sont naturellement modélisés par des classes de WSTS à branchemement infini tels que les réseaux de Petri-ω, c.-à-d. des réseaux de Petri avec des arcs étendus qui permettent de consommer ou créer un nombre arbitraire de jetons.

Nous utilisons des caractérisations algébriques des ensembles clos par le bas [FG09] afin de concevoir des outils mathématiques qui permettent d'établir de nouveaux résultats de décidabilité pour les problèmes de décision des WSTS à branchement infini.

Nous introduisons d'abord une façon de représenter finiment les ensembles clos par le bas à l'instar des bases finies d'ensembles clos par le haut introduits à la section 2.4.2. Par la suite, nous exploitons cette représentation finie afin d'actualiser et de généraliser la notion de complétion de WSTS [FG12], et nous montrons comment elle permet d'analyser finiment les WSTS à branchement infini. Nous

terminons le chapitre par une étude détaillée de la décidabilité et de l'indécidabilité de chacun des problèmes de décision pour les WSTS à branchement infini, en rappelant à chaque fois les résultats connus dans le cas des WSTS à branchement fini.

3.1 Ensembles clos par le bas et idéaux

Il est bien connu que tout sous-ensemble clos par le haut, d'un ensemble muni d'un beau préordre, possède une base finie (section 2.4.2). Or, cela n'est pas toujours le cas des sous-ensembles clos par le bas. Par exemple, considérons $\mathbb{N}$ qui est lui-même clos par le bas. Si $\mathbb{N}$ possédait une base finie, c.-à-d. s'il existait $x_1, x_2, \ldots, x_n \in \mathbb{N}$ tels que $\mathbb{N} = \downarrow\{x_1, x_2, \ldots, x_n\}$, il serait fini, ce qui n'est pas le cas. Un équivalent de la notion de base finie peut néanmoins être retrouvée en introduisant le concept d'idéaux.

Definition 3.1. Soit X un ensemble muni d'un beau préodre. Un *idéal* I de X est un ensemble $\emptyset \subset I \subseteq X$ non vide clos par le bas qui est aussi *dirigé*, c.-à-d.

$$\forall a, b \in I, \exists c \in I \text{ tel que } a \leq c \text{ et } b \leq c \ .$$

Nous dénotons $\mathsf{Ideaux}(X)$ l'ensemble des idéaux de X, c.-à-d. $\mathsf{Ideaux}(X) \stackrel{\text{déf}}{=} \{\emptyset \subset I \subseteq X : I = \downarrow I \text{ et } I \text{ est dirigé}\}$. Notons que $\downarrow x$ est un idéal pour tout $x \in X$.

Nous montrerons que tout sous-ensemble clos par le bas est égal à une union finie d'idéaux. Avant de ce faire, nous explorons les idéaux de $\mathbb{N}^d$ et de Σ^*, et donnons des exemples de telles décompositions.

3.1.1 Idéaux de $\mathbb{N}^d$ et Σ^*

Les idéaux de $\mathbb{N}$ sont décrits de la façon suivante :

Proposition 3.2. $\mathit{Ideaux}(\mathbb{N}) = \{\downarrow x : x \in \mathbb{N}\} \cup \{\mathbb{N}\}$.

Nous remarquons au passage que le seul idéal infini contenu dans $\mathbb{N}$ est $\mathbb{N}$ lui-même. Il existe toutefois une infinité d'idéaux infinis dans $\mathbb{N}^d$ pour $d \geq 2$. Les idéaux de $\mathbb{N}^d$ peuvent s'écrire comme produits cartésiens d'idéaux de $\mathbb{N}$:

Proposition 3.3. *Pour tout $d \in \mathbb{N}_{>0}$,*

$$\mathsf{Ideaux}(\mathbb{N}^d) = \underbrace{\mathsf{Ideaux}(\mathbb{N}) \times \mathsf{Ideaux}(\mathbb{N}) \times \cdots \times \mathsf{Ideaux}(\mathbb{N})}_{d \text{ fois}} .$$

Démonstration. Soit $I \in \mathsf{Ideaux}(\mathbb{N}^d)$. Posons b_i la plus grande valeur qui apparaît en composante i d'un élément de I s'il n'existe qu'un nombre fini de telles valeurs, ou autrement $b_i = \mathbb{N}$. Posons $J = \downarrow b_1 \times \downarrow b_2 \times \ldots \times \downarrow b_d$ et montrons que $I = J$. L'inclusion $I \subseteq J$ est triviale, il suffit donc de montrer que $J \subseteq I$.

Soit $\boldsymbol{x} \in J$, alors pour tout $i \in [d]$, il existe $\boldsymbol{y}_i \in I$ tel que $\boldsymbol{y}_i(i) = b_i$ si b_i est entier, ou tel que $\boldsymbol{y}_i(i) \geq \boldsymbol{x}(i)$ si $b_i = \mathbb{N}$. Puisque I est dirigé, par induction finie, il existe $\boldsymbol{z} \in I$ tel que pour tout $i \in [d]$, $\boldsymbol{y}_i \leq \boldsymbol{z}$. Ainsi, $\boldsymbol{x} \leq \boldsymbol{z}$ et puisque I est clos par le bas, $\boldsymbol{x} \in I$. □

Nous pouvons maintenant donner un exemple de décomposition finie d'un sous-ensemble de $\mathbb{N}^d$ clos par le bas :

Exemple 3.4. Considérons le sous-ensemble clos par le bas suivant :

$$D = \{(x_1, x_2) \in \mathbb{N}^2 : (x_1 \leq 4) \vee (x_1 \leq 8 \wedge x_2 \leq 10) \vee (x_2 \leq 5)\} .$$

Tel qu'illustré à la figure 3.1, il est possible d'écrire D de la façon suivante :

$$\downarrow 4 \times \mathbb{N} \ \cup \ \downarrow 8 \times \downarrow 10 \ \cup \ \mathbb{N} \times \downarrow 5 .$$

Par les propositions 3.2 et 3.3, D est donc égal à une union finie d'idéaux. Notons que ces trois idéaux peuvent être encodés par $\{(4, \omega), (8, 10), (\omega, 5)\}$.

Soit Σ un alphabet fini. Rappelons que l'ordre sous-mot $\preceq$ (section 2.4.1) est un beau préordre pour Σ. Les langages clos par le bas selon cet ordre ont été étudiés

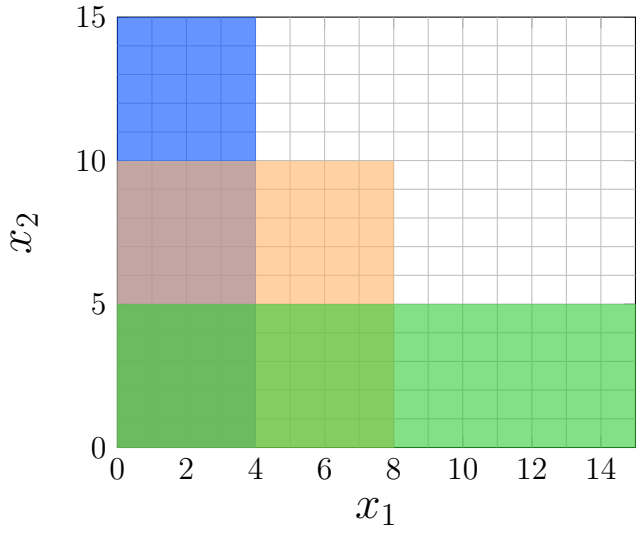


Figure 3.1 – Décomposition de $D = \{(x_1, x_2) \in \mathbb{N}^2 : (x_1 \leq 4) \vee (x_1 \leq 8 \wedge x_2 \leq 10) \vee (x_2 \leq 5)\}$ en un nombre fini d'idéaux. Les trois idéaux $\downarrow 4 \times \mathbb{N}, \downarrow 8 \times \downarrow 10$ et $\mathbb{N} \times \downarrow 5$ apparaissent respectivement en couleur , et .

par Abdulla *et al.* [ACBJ04] dans le but de représenter les ensembles d'accessibilité infinis de systèmes à canaux non fiables. Ces langages ont également été étudiés dans le contexte des langages formels où il a été démontré qu'ils correspondent aux langages strictement testables par morceaux [RHB+09]. Abdulla *et al.* [ACBJ04] ont démontré que tout langage clos par le bas selon $\preceq$ est une union finie de *produits* $P_1 P_2 \cdots P_m$ où chaque P_i est ou bien $\{\varepsilon, \sigma\}$ pour un certain $\sigma \in \Sigma$, ou bien A^* pour un certain $A \subseteq \Sigma$. Il a également été remarqué dans [FG09] que tout idéal $I \in \mathsf{Ideaux}(\Sigma^*)$ s'exprime comme un seul produit :

Proposition 3.5 ([FG09])**.** *Pour tout alphabet fini* Σ*,*

$$\mathsf{Ideaux}(\Sigma^*) = \bigcup_{m \in \mathbb{N}} \{P_1 P_2 \cdots P_m : P_i \in \Sigma \cup \{\varepsilon\} \cup \{A^* : A \subseteq \Sigma\} \text{ pour } i \in [m]\} \ .$$

Les résultats de [ACBJ04] et [FG09] combinés montrent donc que tout langage clos par le bas est égal à une union finie d'idéaux. Nous illustrons ce résultat par un exemple :

Exemple 3.6. Considérons le langage L des mots sur alphabet $\Sigma = \{a, b, c\}$ tels que la première lettre ne réapparait pas, c.-à-d., soit

$$\begin{aligned} L &= \{w \in \Sigma^* \setminus \{\varepsilon\} : w_i \neq w_1 \text{ pour } 1 < i \leq |w|\} \\ &= a\{b, c\}^* \cup b\{a, c\}^* \cup c\{a, b\}^* \ . \end{aligned}$$

Il en découle que

$$\begin{aligned} \downarrow L &= L \cup \{w \in \Sigma^* : |w|_\sigma = 0 \text{ pour un certain } \sigma \in \Sigma\} \\ &= L \cup \{a, b\}^* \cup \{a, c\}^* \cup \{b, c\}^* \\ &= \{a, \varepsilon\}\{b, c\}^* \cup \{b, \varepsilon\}\{a, c\}^* \cup \{c, \varepsilon\}\{a, b\}^* \ . \end{aligned}$$

Ainsi, par la proposition 3.5, $\downarrow L$ ainsi exprimé est une union finie d'idéaux.

3.1.2 Décompositions en idéaux

Avant de démontrer que tout clos par le bas se décompose finiment en idéaux tel qu'illustré aux exemples 3.4 et 3.6, nous prouvons une proposition simple, mais pratique, qui énonce que tout idéal inclus dans une union finie d'idéaux est contenu dans précisément l'un de ces idéaux.

Proposition 3.7. *Soient $I, J_1, J_2, \ldots, J_m \in \mathsf{Ideaux}(X)$ où X est un ensemble muni d'un préordre, alors $I \subseteq J_1 \cup J_2 \cup \cdots \cup J_m$ si, et seulement si, $I \subseteq J_j$ pour un certain $j \in [m]$.*

Démonstration. Nous prétendons d'abord que si un ensemble dirigé I est inclus dans l'union de deux ensembles clos par le bas $J \cup K$, alors ou bien $I \subseteq J$ ou bien $I \subseteq K$. En supposant ce premier énoncé vrai, la proposition peut être démontrée par une induction simple en remarquant que tout idéal est dirigé et que toute union d'idéaux est close par le bas.

Soient $I \subseteq J \cup K$ tels que décrits. Supposons, au contraire, que $I \not\subseteq J$ et $I \not\subseteq K$, alors il existe $s \in I \setminus J$ et $t \in I \setminus K$. Puisque I est dirigé, il existe $u \in I$ tel que

$s \leq u$ et $t \leq u$. Ainsi, puisque $u \in I$, ou bien $u \in J$, ou bien $u \in K$. Puisque J et K sont clos par le bas, cela implique que $s \in J$ ou $t \in K$. Or, cela est une contradiction. □

Nous sommes maintenant en mesure de prouver le théorème principal de cette section.

Théorème 3.8 ([GL13])**.** *Soient X un ensemble muni d'un beau préordre $\leq$, et $D \subseteq X$ un sous-ensemble clos par le bas. Il existe $I_1, I_2, \ldots, I_m \in \mathsf{Ideaux}(X)$ tels que $D = I_1 \cup I_2 \cup \cdots \cup I_m$. De plus, il existe une seule telle collection d'idéaux qui soient deux à deux incomparables, soit, la collection des idéaux maximaux sous l'inclusion dans D.*

Démonstration. Nous disons qu'un sous-ensemble $D \subseteq X$ est *mauvais* s'il est clos par le bas et n'est pas égal à une union finie d'idéaux. Afin d'obtenir une contradiction, supposons qu'il existe un mauvais $D \subseteq X$. Nous pouvons supposer que D est minimal pour l'inclusion parmi l'ensemble des mauvais sous-ensembles puisque les séquences décroissantes de sous-ensembles clos par le bas sont finies (section 2.4.2).

Montrons que D est dirigé. D'abord, $D \neq \emptyset$ puisque l'ensemble vide est égal à une union vide. Ensuite, soient $x_1, x_2 \in D$. Puisque $D \setminus {\uparrow} x_1$ et $D \setminus {\uparrow} x_2$ sont clos par le bas et strictement inclus dans D, ils ne sont pas mauvais, par minimalité de D. Ainsi, $D \setminus {\uparrow} x_1 = \bigcup_{j=1}^{n} I_j$ et $D \setminus {\uparrow} x_2 = \bigcup_{j=n+1}^{m} I_j$ pour certains idéaux $I_1, I_2, \ldots, I_m \in \mathsf{Ideaux}(X)$. Par conséquent,

$$D \setminus ({\uparrow} x_1 \cap {\uparrow} x_2) = (D \setminus {\uparrow} x_1) \cup (D \setminus {\uparrow} x_2) = \bigcup_{j=1}^{m} I_j$$

n'est pas mauvais. Puisque D est mauvais, $D \setminus ({\uparrow} x_1 \cap {\uparrow} x_2)$ est strictement inclus dans D, ce qui implique que $D \cap ({\uparrow} x_1 \cap {\uparrow} x_2)$ est non vide. Soit $y \in D \cap ({\uparrow} x_1 \cap {\uparrow} x_2)$. Nous avons $y \in D$, $x_1 \leq y$ et $x_2 \leq y$. Nous concluons donc que D est dirigé et donc un idéal, ce qui contredit notre hypothèse. Ainsi, D est égal à une union finie d'idéaux.

Supposons qu'il existe deux ensembles distincts $\{I_1, I_2, \ldots, I_m\}$ et $\{J_1, J_2, \ldots, J_n\}$ d'idéaux deux à deux incomparables dont l'union est chacune égale à D. Supposons, sans perte de généralité, que $I_1 \notin \{J_1, J_2, \ldots, J_n\}$. En appliquant la proposition 3.7 deux fois, nous observons que $I_1 \subseteq J_j \subseteq I_i$ pour un certain $j \in [n]$ et un certain $i \in [m]$. Puisque $I_1 \neq J_j$, $I_1 \subsetneq J_j \subseteq I_i$ et ainsi $i \neq 1$. Nous concluons donc que I_1 et I_i sont comparables, ce qui est une contradiction. Cela complète la preuve. □

Remarque 3.9. La preuve de la première moitié du théorème 3.8 donnée dans [FG12] faisait appel à une étude des complétions topologiques et des complétions d'ensembles ordonnés. La preuve présentée ici fut donnée plus tard par Goubault-Larrecq [GL13] et fera partie d'un article en cours d'écriture par un groupe d'auteurs incluant Goubault-Larrecq [GLKK16]. Il fut récemment remarqué par plusieurs chercheurs que ce résultat avait déjà été démontré par Erdős et Tarski en 1943 dans un cadre plus général [ET43], et plus tard par Bonnet [Bon75] et Fraïssé [Fra86].

Afin d'alléger la notation, nous dénoterons la *décomposition en idéaux* d'un sous-ensemble $D \subseteq X$ clos par le bas par $\mathsf{DecompIdeaux}(D)$. Autrement dit,

$$\mathsf{DecompIdeaux}(D) = \max_{\subseteq}\{I \in \mathsf{Ideaux}(X) : I \subseteq D\} \ .$$

Ainsi, le théorème 3.8 affirme que $D = \bigcup_{I \in \mathsf{DecompIdeaux}(D)} I$.

Nous terminons cette section en remarquant que $\mathsf{Ideaux}(X)$ est dénombrable lorsque X est dénombrable.

Proposition 3.10. *Pour tout ensemble X muni d'un beau préordre, $\mathsf{Ideaux}(X)$ est dénombrable si, et seulement si, X est dénombrable.*

Démonstration. Supposons que X soit dénombrable. Tout sous-ensemble clos par le haut possède une unique base finie minimale, ainsi, informellement, il n'y a pas plus de sous-ensembles clos par le bas que de sous-ensembles finis de X, ce qui est dénombrable. Puisque le complément d'un sous-ensemble clos par le bas est un sous-ensemble clos par le haut, et vice versa, les sous-ensembles clos par le bas sont

en bijection avec les sous-ensembles clos par le haut. Puisque les idéaux sont clos par le bas, nous concluons que $\mathsf{Ideaux}(X)$ est dénombrable.

Supposons que $\mathsf{Ideaux}(X)$ soit dénombrable, alors X est dénombrable puisque $x \mapsto {\downarrow} x$ est une fonction injective de X vers $\mathsf{Ideaux}(X)$. □

3.2 Complétions

En exploitant le fait que les ensembles clos par le bas se décomposent finiment en idéaux, nous associons à chaque WSTS un système de transitions ordonné que nous appelons sa complétion. Plutôt que de travailler directement sur les états, la complétion d'un WSTS travaille plutôt sur les idéaux d'états. Bien que la complétion d'un WSTS perde de l'information sur celui-ci, elle préserve un certain nombre de propriétés et de comportements en plus d'avoir l'avantage d'être toujours à branchement fini.

La notion de complétion d'un WSTS a déjà été étudiée dans le cas des WSTS fonctionnels [Fin87a, FG12]. Finkel & Goubault-Larrecq [FG12] définissent la *complétion fonctionnelle* d'un WSTS fonctionnel $\mathcal{S} = (X, \to, F, \leq)$ comme étant le système de transitions fonctionnel et ordonné

$$\overline{\mathcal{S}} = (\mathsf{Ideaux}(X), \to, \overline{F}, \subseteq)$$

où $\overline{F} \stackrel{\text{déf}}{=} \{\overline{f} : f \in F\}$ pour $\overline{f} : \mathsf{Ideaux}(X) \to \mathsf{Ideaux}(X)$ défini par $\overline{f}(I) \stackrel{\text{déf}}{=} {\downarrow} f(I)$. Notons que $\overline{f}$ est bien défini puisque $\overline{f}(I)$ est un idéal. Une preuve élémentaire de ce fait est donné dans la proposition suivante. Ce résultat se retrouve dans [FG12] dans un cadre topologique plus général.

Proposition 3.11. *Soient X un ensemble muni d'un beau préordre, et $f : X \to X$ une fonction partielle non décroissante. Pour tout $I \in \mathsf{Ideaux}(X)$, $\overline{f}(I) \in \mathsf{Ideaux}(X)$.*

Démonstration. Il suffit de montrer que ${\downarrow}\overline{f}(I)$ est dirigé. Soient $a', b' \in {\downarrow}\overline{f}(I)$. Nous avons $a' \leq f(a)$ et $b' \leq f(b)$ pour certains $a, b \in I$. Puisque I est un idéal,

I est dirigé, ainsi il existe $c \in I$ tel que $a \leq c$ et $b \leq c$. De plus, puisque f est non décroissante, nous avons $f(a) \leq f(c)$ et $f(b) \leq f(c)$. Par transitivité, nous obtenons $a' \leq f(c)$ et $b' \leq f(c)$. Puisque $f(c) \in f(I) \subseteq \overline{f}(I)$, nous concluons que $\overline{f}(I)$ est dirigé. □

Nous nous inspirons de la complétion fonctionnelle de Finkel & Goubault-Larrecq afin d'étendre le concept à tout WSTS, incluant les WSTS à branchement infini. Notons que la possibilité d'associer une certaine complétion aux WSTS non fonctionnels avait déjà été observée par Emerson & Namjoshi [EN98].

Definition 3.12. La *complétion* $\widehat{\mathcal{S}}$ d'un WSTS $\mathcal{S} = (X, \to, \Sigma, \leq)$ est le système de transitions ordonné $\widehat{S} = (\mathsf{Ideaux}(X), \rightsquigarrow, \subseteq)$ tel que $I \rightsquigarrow J$ si, et seulement si, $J \in \mathsf{DecompIdeaux}(\downarrow \mathsf{Succ}_{\mathcal{S}}(I))$.

Notons d'abord que cette complétion $\widehat{\mathcal{S}}$ est cohérente avec la complétion fonctionnelle $\overline{\mathcal{S}}$:

Proposition 3.13. *Soit $\mathcal{S} = (X, \to, F, \leq)$ un WSTS fonctionnel. Pour tout $I \in \mathsf{Ideaux}(X)$,*

$$\bigcup_{J \in \mathsf{Succ}_{\overline{\mathcal{S}}}(I)} J = \bigcup_{\overline{f} \in \overline{F}} \overline{f}(I) = \bigcup_{f \in F} \downarrow f(I) = \bigcup_{J \in \mathsf{Succ}_{\widehat{\mathcal{S}}}(I)} J = \downarrow \mathsf{Succ}_{\mathcal{S}}(I).$$

3.2.1 Correspondances entre exécutions

Nous étudions maintenant la relation entre les exécutions d'un WSTS et celles de leur complétion.

Tel qu'annoncé plus tôt, la complétion d'un WSTS $\mathcal{S}$ est à branchement fini, et ce, indépendamment du type de branchement de $\mathcal{S}$:

Proposition 3.14. *$\widehat{\mathcal{S}}$ est à branchement fini pour tout WSTS $\mathcal{S}$.*

Démonstration. Soient $\mathcal{S} = (X, \to, \leq)$ un WSTS et $I \in \mathsf{Ideaux}(X)$. Par définition de $\widehat{\mathcal{S}}$,

$$\mathsf{Succ}_{\widehat{\mathcal{S}}}(I) = \mathsf{DecompIdeaux}(\downarrow \mathsf{Succ}_{\mathcal{S}}(I)) \ .$$

Par le théorème 3.8, cet ensemble est fini, ainsi $\widehat{\mathcal{S}}$ est à branchement fini. □

Dans les deux propositions suivantes, nous mettons en évidence une correspondance entre les exécutions d'un WSTS et celles de sa complétion.

Proposition 3.15. *Soient $\mathcal{S} = (X, \to, \leq)$ un WSTS, et $I, J \in \mathsf{Ideaux}(X)$. Si $I \rightsquigarrow^k J$, alors pour tout $x_J \in J$ il existe $x_I \in I$, $y \geq x_J$ et $k' \in \mathbb{N}$ tels que $x_I \to^{k'} y$. De plus, si $\mathcal{S}$ possède la monotonicité transitive, alors $k' \geq k$; si $\mathcal{S}$ possède la monotonicité forte, alors $k' = k$.*

Démonstration. Soient $\mathcal{S} = (X, \to, \leq)$ un WSTS, et $I, J \in \mathsf{Ideaux}(X)$. Nous procédons par induction sur $k \in \mathbb{N}$.

Si $I \rightsquigarrow^0 J$, alors $I = J$ et pour tout $x_J \in J$, il suffit de prendre $x_I = y = x_J$ et $k' = 0$.

Supposons que $I \rightsquigarrow^{k+1} J$, alors il existe $I' \in \mathsf{Ideaux}(X)$ tel que $I \rightsquigarrow I' \rightsquigarrow^k J$. Par hypothèse d'induction, pour tout $x_J \in J$ il existe $x_{I'} \in I'$, $y' \geq x_J$ et $k' \in \mathbb{N}$ (resp. $k' \geq k$; $k' = k$) tels que $x_{I'} \to^{k'} y'$. Puisque $x_{I'} \in {\downarrow}\mathsf{Succ}_{\mathcal{S}}(I)$, il existe $x_I \in I$ et $y'' \geq x_{I'}$ tels que

$$x_I \to y''. \tag{3.1}$$

De plus, puisque $y'' \geq x_{I'}$, nous pouvons appliquer la monotonicité (resp. transitive ; forte) à $x_{I'} \to^{k'} y'$ et en dériver

$$y'' \to^{k''} y \tag{3.2}$$

pour certains $y \geq y'$ et $k'' \in \mathbb{N}$ (resp. $k'' \geq k'$; $k'' = k'$). Ainsi, par (3.1) et (3.2), nous obtenons l'exécution $x_I \to y'' \to^{k''} y$ de longueur $k''+1$ (resp. $k''+1 \geq k'+1 \geq k+1$; $k''+1 = k'+1 = k+1$), avec $y \geq y' \geq x_J$. □

Proposition 3.16. *Soient $\mathcal{S} = (X, \to, \leq)$ un WSTS, et $x, y \in X$. Si $x \to^k y$, alors pour tout idéal $I \supseteq {\downarrow}x$ il existe un idéal $J \supseteq {\downarrow}y$ tel que $I \rightsquigarrow^k J$.*

Démonstration. Soient $\mathcal{S} = (X, \to, \leq)$ un WSTS, et $x, y \in X$. Nous procédons par induction sur $k \in \mathbb{N}$.

Si $x \to^0 y$, alors $x = y$ et $I \rightsquigarrow^0 I$ pour tout idéal $I \supseteq {\downarrow}x = {\downarrow}y$.

Si $x \to^{k+1} y$, alors il existe $x' \in X$ tel que $x \to x' \to^k y$. Soit I un idéal tel que $I \supseteq \downarrow x$. Nous avons $x' \in \mathsf{Succ}_{\mathcal{S}}(I) \subseteq \downarrow \mathsf{Succ}_{\mathcal{S}}(I)$, ainsi $I \rightsquigarrow I'$ pour un certain idéal $I' \supseteq \{x'\}$. Puisque I' est clos par le bas, nous avons $I' \supseteq \downarrow x'$. Par hypothèse d'induction, il existe un idéal $J \supseteq \downarrow y$ tel que $I' \rightsquigarrow^k J$. Ainsi, $I \rightsquigarrow I' \rightsquigarrow^k J$ est une exécution de longueur $k+1$, ce qui complète la preuve. □

Les propositions 3.15 et 3.16 permettent notamment de démontrer que les ensembles de successeurs de la complétion d'un WSTS $\mathcal{S}$ correspondent précisément aux clôtures par le bas des ensembles de successeurs de $\mathcal{S}$:

Proposition 3.17. *Pour tout WSTS $\mathcal{S} = (X, \to, \leq)$ et tout $x \in X$,*

$$\downarrow \mathsf{Succ}^*_{\mathcal{S}}(x) = \bigcup_{I \in \mathsf{Succ}^*_{\widehat{\mathcal{S}}}(\downarrow x)} I \ .$$

Démonstration. Soit $\mathcal{S} = (X, \to, \leq)$ un WSTS et $x \in X$.

Soit $y \in \mathsf{Succ}^*_{\mathcal{S}}(x)$. En appliquant la proposition 3.16 avec $I = \downarrow x$, nous savons qu'il existe un idéal $J \supseteq \downarrow y$ tel que $J \in \mathsf{Succ}^*_{\widehat{\mathcal{S}}}(\downarrow x)$. Ainsi, $\downarrow y \subseteq \bigcup_{I \in \mathsf{Succ}^*_{\widehat{\mathcal{S}}}(\downarrow x)} I$. Nous avons donc $\downarrow \mathsf{Succ}^*_{\mathcal{S}}(x) \subseteq \bigcup_{I \in \mathsf{Succ}^*_{\widehat{\mathcal{S}}}(\downarrow x)} I$.

Posons $I = \downarrow x$. Soient $J \in \mathsf{Succ}^*_{\widehat{\mathcal{S}}}(I)$ et $x_J \in J$. Par la proposition 3.15, il existe $x_I \in I$, $y \geq x_J$ et $k \in \mathbb{N}$ tel que $x_I \to^k y$. Par définition de I et par monotonicité de $\mathcal{S}$, $x \to^* y'$ pour un certain $y' \geq y$. Ainsi, puisque $x_J \leq y \leq y'$, nous avons $x_J \in \downarrow \mathsf{Succ}^*_{\mathcal{S}}(x)$ et par conséquent $J \subseteq \downarrow \mathsf{Succ}^*_{\mathcal{S}}(x)$. □

3.2.2 Monotonicité et beaux préordres

Il est naturel de se demander si la complétion d'un WSTS est elle aussi un WSTS. Nous remarquons d'abord que toute complétion possède la monotonicité (forte) :

Proposition 3.18. *$\widehat{\mathcal{S}}$ possède la monotonicité forte pour tout WSTS S.*

Démonstration. Soient $I, I', J \in \mathsf{Ideaux}(X)$ tels que $I \rightsquigarrow J$ et $I \subseteq I'$. Nous devons montrer que $I' \rightsquigarrow J'$ pour un certain $J' \in \mathsf{Ideaux}(X)$ tel que $J \subseteq J'$.

Puisque $J \in \mathsf{Succ}_{\widehat{\mathcal{S}}}(I)$ et $I \subseteq I'$, nous avons $J \subseteq {\downarrow}\mathsf{Succ}_{\mathcal{S}}(I) \subseteq {\downarrow}\mathsf{Succ}_{\mathcal{S}}(I')$. Par la proposition 3.7, il existe $J' \in \mathsf{DecompIdeaux}({\downarrow}\mathsf{Succ}_{\mathcal{S}}(I'))$ tel que $J \subseteq J'$. Ainsi, par définition de $\widehat{\mathcal{S}}$, nous concluons que $I' \rightsquigarrow J'$. □

Malheureusement, $\subseteq$ n'est pas toujours un beau préordre pour $\mathsf{Ideaux}(X)$, ainsi la complétion d'un WSTS n'est pas toujours un WSTS. Nous pouvons néanmoins caractériser les WSTS dont la complétion est un WSTS. Cette caractérisation est basée sur la notion de beau préordre dit « ω^2 » étudiée par Jančar [Jan99] :

Definition 3.19. Soit X un ensemble. Nous disons qu'un beau préordre $\leq \subseteq X \times X$ est un *beau préordre-ω^2* si l'ordre de Smyth $\leq^{\#} \subseteq 2^X \times 2^X$ défini par $A \leq^{\#} B \Leftrightarrow {\uparrow}B \subseteq {\uparrow}A$ est aussi un beau préordre.

Il est connu que $\subseteq$ est un beau préordre pour $\mathsf{Ideaux}(X)$ si, et seulement si, $\leq$ est un beau préordre-ω^2 pour X (p. ex. voir [FG12, proposition 4.3]). Bien que les beaux préordres ne soient pas tous ω^2, la plupart des ensembles d'états de WSTS qui apparaissent dans la littérature le sont. Le contre-exemple typique [Rad54] de beau préordre qui n'est pas ω^2 est l'*ordre de Rado* $\leq_{\mathrm{Rado}}$ défini sur $X_{\mathrm{Rado}} \stackrel{\text{déf}}{=} \{(m,n) \in \mathbb{N}^2 : m < n\}$ par

$$(m,n) \leq_{\mathrm{Rado}} (m',n') \;\Leftrightarrow\; (m = m' \wedge n \leq n') \vee (n < m') \;.$$

Il est bien connu que $\leq_{\mathrm{Rado}}$ est un beau préordre, et que $\leq^{\#}_{\mathrm{Rado}}$ n'est pas un beau préordre pour $2^{X_{\mathrm{Rado}}}$ [Jan99].

Ces résultats nous permettent d'étendre la terminologie aux WSTS.

Definition 3.20 ([FG12])**.** Nous disons qu'un WSTS $\mathcal{S} = (X, \to, \leq)$ est un *WSTS-ω^2* lorsque $\leq$ est un beau préordre-ω^2.

Grâce à la proposition 3.18, nous obtenons ainsi une généralisation du résultat suivant connu pour les WSTS fonctionnels [FG12] :

Théorème 3.21. *Pour tout WSTS $\mathcal{S}$, $\widehat{\mathcal{S}}$ est un WSTS si, et seulement si, $\mathcal{S}$ est un WSTS-ω^2.*

Afin de conclure nos observations sur la monotonicité et les beaux préordres, nous remarquons qu'un WSTS « hérite » toujours de la monotonicité stricte de sa complétion, alors que la réciproque n'est pas nécessairement vraie.

Proposition 3.22. *Soit $\mathcal{S} = (X, \to, \leq)$ un WSTS. Si $\widehat{\mathcal{S}}$ possède la monotonicité stricte, alors $\mathcal{S}$ la possède également. Toutefois, il existe un WSTS $\mathcal{S}$ qui possède la monotonicité stricte, alors que $\widehat{\mathcal{S}}$ ne la possède pas.*

Démonstration. Soit $\mathcal{S} = (X, \to, \leq)$ un WSTS tel que $\widehat{S}$ possède la monotonicité stricte. Soient $x, x', y \in X$ tels que $x < x'$ et $x \to y$. Nous devons montrer que $x' \to^* y'$ pour un certain $y' \in X$ tel que $y < y'$.

Puisque $y \in \mathsf{Succ}_{\mathcal{S}}(x)$, nous avons $\downarrow y \subseteq \downarrow \mathsf{Succ}_{\mathcal{S}}(\downarrow x)$. Ainsi, par la proposition 3.7, nous avons $\downarrow y \subseteq J$ pour un certain $J \in \mathsf{DecompIdeaux}(\downarrow \mathsf{Succ}_{\mathcal{S}}(\downarrow x))$. Par définition de $\widehat{\mathcal{S}}$, nous avons donc $\downarrow x \rightsquigarrow J$.

Puisque $\downarrow x \subset \downarrow x'$, il existe $J' \in \mathsf{Ideaux}(X)$ tel que $\downarrow x' \rightsquigarrow^* J'$ et $J \subset J'$ par monotonicité stricte de $\widehat{\mathcal{S}}$. Puisque $J \subseteq J'$, nous avons $y \in J'$. Soit $b \in J' \setminus J$, alors il existe $c \in J'$ tel que $y \leq c$ et $b \leq c$ puisque J' est dirigé. Notons que $c \notin J$, autrement cela impliquerait que $b \in J$ puisque J est clos par le bas. De plus, $c > y$ car $c = y$ impliquerait $c \in J$.

Par la proposition 3.15, il existe $x'' \in \downarrow x'$ et $c' \geq c$ tels que $x'' \to^* c'$. Par monotonicité (standard), nous avons $x' \to^* y'$ pour un certain $y' \geq c'$. Puisque $y' \geq c' \geq c > y$, cela montre que S possède la monotonicité stricte.

Afin de montrer que la réciproque n'est pas toujours vraie, posons $\mathcal{S} = (\mathbb{N}^2, \to, \leq)$ le WSTS tel que $(a, b) \to (0, a + b)$. Notons que $\mathcal{S}$ possède la monotonicité stricte. Posons $I = \mathbb{N} \times \downarrow 0$ et $I' = \mathbb{N} \times \downarrow 1$. Nous avons $I \subset I'$, mais $\mathsf{Succ}_{\widehat{\mathcal{S}}}(I) = \mathsf{Succ}_{\widehat{\mathcal{S}}}(I') = \{\downarrow 0 \times \mathbb{N}\}$. Nous concluons donc que $\widehat{\mathcal{S}}$ ne possède pas la monotonicité stricte. □

3.2.3 Classes et effectivité

Les complétions de WSTS s'avéreront utiles à la section suivante pourvu qu'il soit possible de les manipuler. Nous étendons donc les notions d'effectivité (sec-

tion 2.8.4). Soit $\mathcal{C}$ une classe de WSTS, nous définissons la *complétion* de $\mathcal{C}$ comme étant la classe $\widehat{\mathcal{C}}$ de systèmes de transitions ordonnés définie par $\widehat{\mathcal{C}}(i) \stackrel{\text{déf}}{=} \widehat{\mathcal{C}(i)}$. Nous supposons que les états de $\widehat{\mathcal{C}}$ sont manipulables comme défini à la section 2.8.4. Nous rappelons les définitions pertinentes afin de faciliter la tâche du lecteur.

Soit X_i l'ensemble des états de $\mathcal{C}(i)$. La classe $\widehat{\mathcal{C}}$ est associée à une fonction surjective $\mathsf{repr}_{\widehat{\mathcal{C}}} : \mathbb{N} \times \mathbb{N} \to \bigcup_i \mathsf{Ideaux}(X_i)$ telle que l'ensemble $\{(i,e) : \mathsf{repr}_{\widehat{\mathcal{C}}}(i,e) \in \mathsf{Ideaux}(X_i)\}$ soit décidable. Nous définissons $\mathsf{enc}_{\widehat{\mathcal{C}}} : \mathbb{N} \to 2^{\mathbb{N}}$ par

$$\mathsf{enc}_{\widehat{\mathcal{C}}}(i) \stackrel{\text{déf}}{=} \{e \in \mathbb{N} : \mathsf{repr}_{\widehat{\mathcal{C}}}(i,e) \in \mathsf{Ideaux}(X_i)\} \ .$$

Nous disons que $e \in \mathbb{N}$ est un *encodage* de $I \in \mathsf{Ideaux}(X_i)$ si $\mathsf{repr}_{\widehat{\mathcal{C}}}(i,e) = I$.

Nous avons vu à la sous-section 3.2.1 que les exécutions d'un WSTS sont reliées à celles de sa complétion. Pour exploiter ces relations algorithmiquement, nous faisons l'hypothèse qu'il existe une façon de relier effectivement les états X d'un WSTS à ses idéaux $\mathsf{Ideaux}(X)$. Dans le but d'appliquer les propositions 3.15 et 3.16, nous imposons l'existence d'une procédure qui permet d'obtenir l'encodage de l'idéal $\downarrow x$ à partir d'un état x. Cette contrainte sera suffisante pour obtenir nos résultats de décidabilité. Par contre, il semble nécessaire de pouvoir manipuler les idéaux qui ne sont pas de la forme $\downarrow x$, p. ex. l'idéal $\mathbb{N}^d \in \mathsf{Ideaux}(\mathbb{N}^d)$, lorsque l'on cherche à formaliser l'indécidabilité de problèmes. Puisque tout idéal s'exprime comme le complément d'un ensemble clos par le haut, et que tout ensemble clos par le haut possède une base finie minimale, nous faisons l'hypothèse qu'il est possible d'obtenir l'encodage des idéaux maximaux contenus dans $\overline{\uparrow\{x_1, x_2, \ldots, x_m\}}$. Par exemple, l'encodage de $\mathbb{N}^d \in \mathsf{Ideaux}(\mathbb{N}^d)$ peut ainsi être obtenu grâce à $\overline{\uparrow\emptyset}$. Cette notion d'effectivité est définie formellement de la façon suivante :

Definition 3.23. Soit $\mathcal{C}$ une classe de WSTS possédant les propriétés suivantes :

(a) il existe une machine de Turing $M_\downarrow$ qui retourne, sur entrée (i,x) où $i \in \mathbb{N}$ et x est un état de $\mathcal{C}(i)$, un encodage de $\downarrow x$, c.-à-d., $e \in \mathsf{enc}_{\widehat{\mathcal{C}}}(i)$ tel que $\mathsf{repr}_{\mathcal{C}}(i,e) = \downarrow x$;

(b) il existe une machine de Turing $M_{\uparrow^c}$ qui retourne, sur entrée $(i, \{x_1, x_2, \ldots, x_m\})$ où $i \in \mathbb{N}$ et $\{x_1, x_2, \ldots, x_m\}$ est un ensemble d'états de $\mathcal{C}(i)$, un encodage d'une base minimale de $\overline{\uparrow\{x_1, x_2, \ldots, x_m\}}$, c.-à-d. $e_1, e_2, \ldots, e_n \in \mathsf{enc}_{\widehat{\mathcal{C}}}(i)$ tels que

$$\mathsf{DecompIdeaux}(\overline{\uparrow\{x_1, x_2, \ldots, x_m\}}) = \{\mathsf{repr}_{\mathcal{C}}(i, e_1), \mathsf{repr}_{\mathcal{C}}(i, e_2), \ldots, \mathsf{repr}_{\mathcal{C}}(i, e_n)\} \ .$$

Nous disons d'une telle classe $\mathcal{C}$ qu'elle est *effective sous complétion* (resp. *post-effective sous complétion* si la classe $\widehat{\mathcal{C}}$ est effective (resp. post-effective). Par extension, nous disons qu'un WSTS $\mathcal{S}$ est *effectif sous complétion* (resp. *post-effectif sous complétion*) est la classe dégénérée $\{\mathcal{S}\}$ est effective sous complétion (resp. post-effective sous complétion).

Nous remarquons que la (post-)effectivité d'une classe $\mathcal{C}$ de WSTS est indépendante de la (post-)effectivité de sa complétion $\widehat{\mathcal{C}}$. En fait, cela est même vrai pour les classes dégénérées de WSTS tel que démontré dans les deux propositions suivantes.

Proposition 3.24. *Il existe un WSTS post-effectif qui n'est pas effectif sous complétion.*

Démonstration. Nous donnons une preuve inspirée de [FMP04, proposition 2.4]. Soit $\mathcal{S} = (\mathbb{N}^2, \to, \{g\} \leq)$ le WSTS fonctionnel tel que

$$g(m, n) = (m + |\{i \leq m \ : \mathsf{Turing}_i \text{ s'arrête sur son encodage en au plus } n \text{ étapes}\}|, n) \ .$$

Nous remarquons que $\mathcal{S}$ est post-effectif puisque $\to$ peut être calculé en simulant un certain nombre de machines de Turing pendant un nombre fini d'étapes, et puisque $|\mathsf{Succ}_{\mathcal{S}}(\boldsymbol{x})| = 1$ pour tout $\boldsymbol{x} \in \mathbb{N}^2$.

Montrons que $\widehat{\mathcal{S}}$ n'est pas effective. Soit I_m l'idéal $\downarrow m \times \mathbb{N}$, et J_m l'unique idéal tel que $I_m \rightsquigarrow J_m$. Une induction simple sur m montre que pour tout $m > 0$, il

existe $a \in \mathbb{N}$ tel que $J_{m-1} = \downarrow a \times \mathbb{N} = I_a$ et

$$J_m = \begin{cases} \downarrow (a+2) \times \mathbb{N} & \text{si } M_m \text{ s'arrête sur son encodage,} \\ \downarrow (a+1) \times \mathbb{N} & \text{sinon.} \end{cases}$$

Supposons que $\widehat{\mathcal{S}}$ soit effectif. Montrons que cela permet de décider si M_m s'arrête ou non sur son encodage. Par éfinition, il est possible de calculer l'encodage de I_i en calculant $\mathsf{DecompIdeaux}(\overline{\uparrow\{(i+1,0)\}})$ qui est égal à $\downarrow i \times \mathbb{N}$. Ainsi, nous pouvons calculer I_{m-1} et I_m, ce qui nous permet de calculer J_{m-1} et J_m grâce au fait que $\rightsquigarrow$ est calculable. Nous savons que $J_{m-1} = I_a = \downarrow a \times \mathbb{N}$ pour un certain $a \in \mathbb{N}$. Tel que mentionné, il est possible d'obtenir l'encodage de $I_0, I_1, \ldots$ et, ainsi, afin de déterminer a, nous testons $I_i \subseteq J_{m-1}$ pour $i \geq 0$ jusqu'à ce que nous atteignions $I_{a+1} \not\subseteq J_{m-1}$. Notons qu'il est possible de tester $I_i \subseteq J_{m-1}$ car $\widehat{\mathcal{S}}$ est effectif. De façon similaire, nous savons que $J_m = I_{a+b}$ pour un certain $b \in \{1, 2\}$ et nous pouvons trouver la valeur de b avec la même stratégie. Si $b = 1$, alors M_m s'arrête sur son encodage, sinon elle ne s'arrête pas. Nous concluons donc que si $\widehat{\mathcal{S}}$ était effective, nous pourrions décider le problème d'arrêt, ce qui est impossible. □

Proposition 3.25. *Il existe un WSTS post-effectif sous complétion qui n'est pas effectif.*

Démonstration. Soit $\mathcal{S} = (\mathbb{N}_\omega, \to, \leq)$ le WSTS défini par

- $i \to j$ si $i, j \in \mathbb{N}$ et Turing_i s'exécute sur son encodage pendant plus de j étapes,
- $i \to \omega$ pour tout $i \in \mathbb{N}_\omega$.

Il est assez simple de voir que $\mathcal{S}$ est effectif. Cependant, $\mathcal{S}$ n'est pas post-effectif. En effet, $\mathsf{Succ}_{\mathcal{S}}(i)$ est fini si, et seulement si, Turing_i s'arrête sur son encodage. Ainsi, si S était post-effectif, il existerait une machine de Turing capable de décider le problème d'arrêt, ce qui serait une contradiction.

Toutefois, $\widehat{\mathcal{S}}$ est post-effectif car $\mathsf{Succ}_{\widehat{\mathcal{S}}}(I) = \{\mathbb{N}_\omega\}$ pour tout $I \in \mathsf{Ideaux}(\mathbb{N}_\omega)$. □

3.2.4 Exemples de classes post-effectives sous complétion

Tel que mentionné plus tôt, la notion de classes (post-)effectives sous complétion se retrouvera au coeur de la décidabilité des différents problèmes étudiés à la prochaine section. Pour donner une certaine intuition et se convaincre que la notion d'effectivité sous complétion est naturelle, nous montrons que les réseaux affines, les systèmes d'addition de vecteurs, les réseaux de Petri et les réseaux de Petri-ω sont tous post-effectifs sous complétion. La plupart des autres modèles de WSTS sont également post-effectifs sous complétion, p. ex. voir [ABJ98, BFM16] pour le cas des systèmes à canaux non fiables qui travaillent sur les mots plutôt que $\mathbb{N}^d$.

Exemple 3.26 (Réseaux affines, VASS et réseaux de Petri)**.** Soient (Q, T) un réseau affine à d compteurs et $\mathcal{S} = (Q \times \mathbb{N}^d, \to, F, \leq)$ le WSTS qui lui est associé. Tel que noté à l'exemple 3.4, les éléments de $\mathsf{Ideaux}(\mathbb{N}^d)$ peuvent être représentés par les éléments de $\mathbb{N}_\omega^d$. Cela est non seulement pratique pour l'encodage, mais permet également d'étendre naturellement les fonctions affines induites par T de $Q \times \mathbb{N}^d$ à $Q \times \mathbb{N}_\omega^d$. Soient $t \in T$ et $f_t \in F$ la fonction partielle associée à t, nous étendons f_t à $Q \times \mathbb{N}_\omega^d$ avec la règle $\omega + x = x + \omega = \omega$ pour chaque $x \in \mathbb{N}_\omega^d$. Posons $\widehat{F} = \{\widehat{f}_t : f_t \text{ étendue à } Q \times \mathbb{N}_\omega^d\}$. La complétion de $\mathcal{S}$ peut être représentée par

$$\widehat{\mathcal{S}} = (Q \times \mathbb{N}_\omega^d, \rightsquigarrow, \leq)$$

où $I \rightsquigarrow J$ si $J \in \max\{\widehat{f}(I) : \widehat{f} \in \widehat{F}\}$ et $\leq$ est l'ordre obtenu pour $Q \times \mathbb{N}_\omega^d$ tel que décrit à la section 2.4.

Puisque $\leq$ est calculable (en temps polynomial) pour $Q \times \mathbb{N}_\omega^d$, il est possible de calculer les éléments de $\max\{\widehat{f}(I) : \widehat{f} \in \widehat{F}\}$, et ainsi de calculer $\rightsquigarrow$ et $\mathsf{Succ}_{\widehat{\mathcal{S}}}(I)$, en appliquant la proposition 3.7, c.-à-d. en comparant les éléments de $\{f(I) : f \in \widehat{F}\}$ deux à deux afin d'obtenir les idéaux maximaux.

Exemple 3.27 (Réseaux de Petri-ω)**.** Un réseau de Petri-ω est un réseau de Petri dont certains arcs peuvent être étiquetés par ω plutôt que par un entier. Un tel arc entrant (resp. sortant) consomme (resp. crée) un nombre entier non négatif, mais

arbitraire, de jetons. Afin de montrer que les réseaux de Petri-ω sont post-effectifs sous complétion, nous étendons leurs transitions aux idéaux et fermons leur image par le bas.

Soient $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri-ω, et $\mathcal{S} = (\mathbb{N}^P, \to, T, \leq)$ le WSTS qui lui est associé. Soient $p \in P$ une place et $t \in T$ une transition de $\mathcal{N}$ telles que $\mathbf{Pre}(t,p) = a \in \mathbb{N}_\omega$ et $\mathbf{Post}(p,t) = b \in \mathbb{N}_\omega$. Nous représentons $\mathsf{Ideaux}(\mathbb{N}^P)$ par $\mathbb{N}_\omega^P$, et nous étendons le comportement de t à $\mathbb{N}_\omega^P$. Soit $I \in \mathbb{N}_\omega^P$. Si $a = b = 0$, alors t laisse $I(p)$ inchangé. Supposons que $a \neq 0$ ou $b \neq 0$. Si $I(p) = \omega$, alors t envoie $I(p)$ sur ω ; si $I(p) = n$ pour un certain $n \in \mathbb{N}$, alors t envoie $I(p)$ sur

- $n - a + b$ si $a \in \mathbb{N}$, $b \in \mathbb{N}$, et $n \geq a$,
- $n + b$ si $a = \omega$, $b \in \mathbb{N}$,
- ω si $a \in \mathbb{N}$, $b = \omega$, et $n \geq a$,
- ω si $a = \omega$, $b = \omega$,
- « non défini » sinon.

Pour obtenir $\mathsf{Succ}_{\widehat{\mathcal{S}}}(I)$, il suffit de calculer

$$\max\{J : J \text{ est obtenu en activant une transition } t \text{ à partir de } I\}$$

en appliquant la proposition. 3.7.

Dans les deux exemples ci-dessus, nous n'avons pas montré comment satisfaire les points (a) et (b) de la définition 3.23 pour $\mathbb{N}^d$. L'encodage de l'idéal $\downarrow \boldsymbol{x}$ à partir de $\boldsymbol{x} \in \mathbb{N}^d$ est direct. Cependant, l'obtention d'une décomposition de $\overline{\uparrow\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}}$ en idéaux est plus subtile. Soient $j \in [d]$ et $a > 0$, posons $\boldsymbol{e}_{j,a} \in \mathbb{N}_\omega^d$ le vecteur qui prend la valeur ω dans toutes ses composantes, à l'exception de la composante j qui prend la valeur a. Soit $i \in [n]$, posons $Y_i \stackrel{\text{déf}}{=} \{\boldsymbol{e}_{j,\boldsymbol{x}_i(j)-1} : j \in [d], \boldsymbol{x}_i(j) > 0\}$. Nous dénotons $\phi : \mathbb{N}_\omega^d \to \mathsf{Ideaux}(\mathbb{N}_\omega^d)$ la

bijection naturelle entre $\mathbb{N}_\omega^d$ et $\mathsf{Ideaux}(\mathbb{N}_\omega^d)$. Nous avons

$$\begin{aligned}
\overline{\uparrow\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}} &= \overline{\uparrow \boldsymbol{x}_1 \cup \uparrow \boldsymbol{x}_2 \cup \cdots \cup \uparrow \boldsymbol{x}_n} \\
&= \bigcap_{i=1}^{n} \overline{\uparrow \boldsymbol{x}_i} \\
&= \bigcap_{i=1}^{n} \bigcup_{\boldsymbol{y} \in Y_i} \phi(\boldsymbol{y}) \\
&= \bigcup_{(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n) \in Y_1 \times Y_2 \times \cdots \times Y_n} \bigcap_{i \in [n]} \phi(\boldsymbol{y}_i) \\
&= \bigcup_{(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n) \in Y_1 \times Y_2 \times \cdots \times Y_n} \phi(\mathsf{minvec}(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n))
\end{aligned}$$

où $\mathsf{minvec}(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n) \in \mathbb{N}_\omega^d$ est le vecteur tel que, pour tout $j \in [d]$,

$$\mathsf{minvec}(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n)(j) \stackrel{\text{déf}}{=} \min(\boldsymbol{y}_1(j), \boldsymbol{y}_2(j), \ldots, \boldsymbol{y}_n(j)) \ .$$

Puisque les ensembles $Y_1, Y_2, \ldots, Y_n$ et $\mathsf{minvec}(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n)$ sont calculables en temps polynomial, cela mène à une représentation effective de $\overline{\uparrow\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}}$. Pour obtenir la décomposition minimale, il suffit d'appliquer la proposition 3.7.

3.3 Décidabilité et indécidabilité

Nous sommes maintenant en mesure d'étudier la décidabilité des problèmes de décision de la section 2.8.5 pour les WSTS, à l'exception du problème d'accessibilité qui est indécidable pour essentiellement toutes hypothèses raisonnables. Nous dédions à chaque problème une sous-section dans laquelle nous identifions la frontière séparant décidabilité et indécidabilité dans les cas à branchement fini et infini.

Le lemme suivant nous sera utile pour raisonner sur les arbres infinis :

Lemme de König ([Fra97])**.** Soit T un arbre infini possédant une racine r. Si T est à branchement fini, c'est-à-dire que chaque sommet de T est adjacent à un nombre fini de sommets, alors il existe un chemin infini à partir de r dans T.

3.3.1 Problèmes de terminaison et de terminaison forte

Rappelons que le problème de terminaison consiste à déterminer s'il n'existe pas d'exécution infinie à partir d'un état initial x, alors que le problème de terminaison forte consiste à déterminer si toutes les exécutions à partir de x sont bornées par une certaine longueur. Nous remarquons que, pour les WSTS à branchement fini, ces deux problèmes coïncident :

Proposition 3.28. *Soient $\mathcal{S} = (X, \to)$ un système de transitions et $x \in X$. Il n'existe pas d'exécution infinie à partir de x si, et seulement si, il existe $m \in \mathbb{N}$ tel que $y \in \mathsf{Exec}(x) \implies |y| \leq m$.*

Démonstration. Soient $\mathcal{S} = (X, \to, \leq)$ un WSTS et $x \in X$. Supposons qu'il n'existe pas d'exécution infinie à partir de x. L'arbre d'accessibilité $T(x)$ induit par $\to$ à partir de x est donc fini. Il suffit donc de prendre m comme étant la longueur de la plus longue branche de $T(x)$.

Supposons qu'il existe $m \in \mathbb{N}$ tel que $y \in \mathsf{Exec}(x) \implies |y| \leq m$. Afin d'obtenir une contradiction, supposons qu'il existe une exécution infinie à partir de x. Dans ce cas, l'arbre d'accessibilité $T(x)$ est forcément infini. Par le lemme de König, T possède une branche infinie. Or, cela contredit l'existence de m qui borne la longueur de chaque branche. □

Il est bien connu que dans le cas à branchement fini, le problème de terminaison est décidable en exploitant les propriétés de la monotonicité transitive. Ainsi, par la proposition 3.28, cela s'applique aussi au problème de terminaison forte :

Théorème 3.29 ([FPS01])**.** *Le problème de terminaison et le problème de terminaison forte sont décidables pour toute classe post-effective $\mathcal{C}$ de WSTS à branchement fini avec monotonicité transitive.*

Lorsque la monotonicité n'est plus transitive, nous observons que le problème de terminaison devient indécidable :

Théorème 3.30. *Il existe une classe post-effective de WSTS-ω^2 à branchement fini avec bel ordre pour laquelle le problème de terminaison et le problème de terminaison forte sont indécidables.*

Démonstration. Soit $\mathcal{C}(i) \stackrel{\text{déf}}{=} (\mathbb{N}, \to, \leq)$ le système de transition défini par $x \to x+1$ lorsque Turing_i ne s'arrête pas sur son encodage en x étapes ou moins.

Montrons que $\mathcal{C}$ est une classe post-effective de WSTS à branchement fini. Puisque chaque état possède au plus un successeur, $\mathcal{C}(i)$ est à branchement fini. De plus, $\mathcal{C}$ est post-effective car $\to$ et $|\mathsf{Succ}_{\mathcal{C}(i)}(x)|$ peuvent être calculés en exécutant Turing_i pendant un nombre fini d'étapes. Puisque $\leq$ est un beau préordre, il suffit donc de montrer que $\mathcal{C}(i)$ est monotone. Soient $x, x', y \in \mathbb{N}$ tels que $x \leq x'$ et $x \to y$. Par définition de $\to$, nous avons $y = x + 1$. Si $x = x'$, alors $x' \to x + 1$ et nous avons terminé puisque $y \leq x + 1$. Si $x < x'$, alors trivialement $x' \to^* x'$ et nous avons aussi terminé puisque $y = x + 1 \leq x'$.

Afin de conclure la preuve, notons qu'il existe une exécution infinie à partir de 0 dans $\mathcal{C}(i)$ si, et seulement si, Turing_i ne s'arrête pas sur son encodage. □

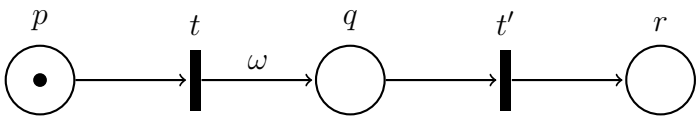


Figure 3.2 – Exemple de réseau de Petri-ω qui termine à partir de $\boldsymbol{x} = (1, 0, 0)$, mais qui ne termine pas fortement.

La preuve de la proposition 3.28 dépend fortement du lemme de König. En effet, le problème de terminaison et le problème de termaison forte diffèrent subtilement lorsque le branchement des WSTS est infini. Par exemple, considérons le réseau de Petri-ω illustré à la figure 3.2. Considérons le marquage $\boldsymbol{x} = (1, 0, 0)$, marquant respectivement les places p, q et r. Tel qu'illustré à la figure 3.3, initialement, seule la transition t peut être activée en menant ainsi au marquage $(0, m, 0)$ pour un certain $m \in \mathbb{N}$. Après coup, seule la transition t' peut être déclenchée, et ce au plus

m fois. Ainsi, il n'existe pas d'exécution infinie à partir de $\boldsymbol{x}$ bien qu'il n'existe aucune borne sur la longueur des exécutions.

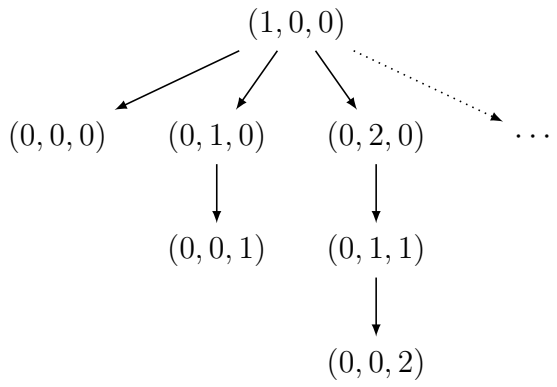


Figure 3.3 – Arbre d'accessibilité, à partir de $\boldsymbol{x} = (1, 0, 0)$, du réseau de Petri-$\omega$ illustré à la figure 3.2.

Nous montrons que contrairement au cas à branchement fini, le problème de terminaison est indécidable sous les hypothèses les plus fortes introduites dans cette thèse :

Théorème 3.31. *Il existe une classe, post-effective et post-effective sous complétion, de WSTS-ω^2 avec monotonicité forte et stricte, et bel ordre, pour laquelle le problème de terminaison est indécidable.*

Démonstration. Nous savons de [DJPS99] que le problème de terminaison structurelle, c.-à-d. le problème qui consiste à déterminer si pour tout état x il n'existe pas d'exécution infinie à partir de x, est indécidable pour les réseaux de Petri avec transferts. Les réseaux de Petri avec transferts forment une classe $\mathcal{C}$ de WSTS vérifiant les propriétés de l'énoncé du théorème. Nous montrons que le problème de terminaison structurelle se réduit au problème de terminaison pour une classe $\mathcal{D}$ vérifiant les mêmes propriétés.

Soit $\mathcal{C}(i) = (X, \rightarrow, \leq)$ où $X = \mathbb{N}^P$ et P est un ensemble fini de places. Nous définissons

$$\mathcal{D}(i) \stackrel{\text{déf}}{=} (X \cup \{x_0\}, \rightarrow \cup \{(x_0, x) : x \in X\}, \leq \cup \{(x_0, x_0)\})$$

comme étant le système de transitions ordonné obtenu à partir de $\mathcal{C}(i)$ en ajoutant un nouvel élément « initial » x_0. Nous remarquons que $\mathcal{D}(i)$ est aussi un WSTS-ω^2 avec monotonicité forte et stricte, muni d'un bel ordre. Notons que $\mathcal{D}(i)$ est à branchement infini puisque $\mathsf{Succ}_{\mathcal{D}(i)}(x_0) = X$.

Nous montrons que la classe $\mathcal{D}$ est aussi post-effective et post-effective sous complétion. En effet, x_0 peut être encodé par le symbole spécial $\#$; l'unique nouvel idéal est $\{x_0\}$ qui peut aussi être encodé par un nouveau symbole \$; et les machines de Turing qui calculent $\to$, $\leq$, $\rightsquigarrow$ et $\subseteq$ sont aisément adaptées afin de gérer ces deux nouveaux symboles.

Afin de conclure la preuve, nous remarquons qu'il n'existe pas d'exécution infinie dans $\mathcal{C}(i)$ si, et seulement si, il n'existe pas d'exécution infinie à partir de x_0 dans $\mathcal{D}(i)$. □

En revanche, nous montrons que le problème de terminaison forte est décidable pour les classes de WSTS avec monotonicité transitive, comme dans le cas à branchement fini, pourvu que leur complétion satisfasse quelques propriétés additionnelles :

Théorème 3.32. *Le problème de terminaison forte est décidable pour toute classe, post-effective sous complétion, de WSTS-ω^2 avec monotonicité transitive.*

Démonstration. Soit $\mathcal{C}$ une classe vérifiant les propriétés de l'énoncé du théorème. Soit $\mathcal{S} \in \mathcal{C}$, puisque $\mathcal{S}$ est un WSTS-ω^2, $\widehat{\mathcal{S}}$ est un WSTS. De plus, $\widehat{\mathcal{S}}$ possède la monotonicité forte par la proposition 3.18 et ainsi la monotonicité transitive. Puisque $\mathcal{C}$ est effective sous complétion, $\widehat{\mathcal{C}}$ est post-effective. Ainsi, $\widehat{\mathcal{C}}$ est une classe post-effective de WSTS à branchement fini avec monotonicité transitive. Par le théorème 3.29, le problème de terminaison forte est donc décidable pour $\widehat{\mathcal{C}}$.

Montrons que cela implique la décidabilité du problème pour $\mathcal{C}$. Soit $\mathcal{S} = (X, \to, \leq) \in \mathcal{C}$. Nous prétendons qu'il existe une borne sur la longueur des exécutions à partir de $x \in X$ dans $\mathcal{S}$ si, et seulement si, il existe une borne sur la longueur des exécutions à partir de $\downarrow x$ dans $\widehat{\mathcal{S}}$. Ainsi, il est possible de décider la terminaison forte à partir de x dans $\mathcal{S}$ en la décidant plutôt à partir de $\downarrow x$ dans $\widehat{\mathcal{S}}$. Puisque $\mathcal{C}$

est post-effective sous complétion, cela est possible en obtenant l'encodage de $\downarrow x$ à partir de x.

Supposons qu'il existe une borne $m \in \mathbb{N}$ sur la longueur des exécutions à partir de x dans $\mathcal{S}$. Afin d'obtenir une contradiction, supposons que $\downarrow x \rightsquigarrow^{m'} J$ pour un certain $m' > m$ et un certain $J \in \mathsf{Ideaux}(X)$. Par la proposition 3.15, il existe $x' \in \downarrow x$ et $y' \in X$ tels que $x' \rightarrow^{k'} y'$ où $k' \geq m'$. Par monotonicité transitive, $x \rightarrow^{k} y'$ pour un certain $y \geq y'$ et $k \geq k'$. Ainsi, il existe une exécution de longueur $k \geq k' \geq m' > m$ à partir de x, ce qui est une contradiction.

De façon réciproque, supposons qu'il existe une borne $m \in \mathbb{N}$ sur la longueur des exécutions à partir de $\downarrow x$ dans $\widehat{\mathcal{S}}$. Afin d'obtenir une contradiction, supposons que $x \rightarrow^{m'} J$ pour un certain $m' > m$ et un certain $y \in X$. Par la proposition 3.16, il existe $J \in \mathsf{Ideaux}(X)$ tel que $\downarrow x \rightsquigarrow^{m'} J$. Cela est une contradiction, ce qui complète la preuve. □

3.3.2 Problème de finitude

Le problème de finitude consiste à déterminer si l'ensemble des successeurs $\mathsf{Succ}^*(x)$ d'un état x est fini. La décidabilité de ce problème est connue pour les classes de WSTS à branchement fini :

Théorème 3.33 ([Fin87a, FPS01, DFPS98])**.** *Le problème de finitude est décidable pour toute classe post-effective $\mathcal{C}$ de WSTS à branchement fini avec monotonicité transitive et stricte, et bel ordre. De plus, il existe une classe, post-effective et post-effective sous complétion, $\mathcal{D}$ de WSTS-ω^2 à branchement fini avec monotonicité forte et bel ordre pour laquelle le problème est indécidable.*

Nous généralisions le théorème précédent en retirant deux hypothèses : la monotonicité transitive et le branchement fini.

Théorème 3.34. *Le problème de finitude est décidable pour toute classe post-effective de WSTS avec monotonicité stricte et bel ordre.*

Démonstration. Soit $\mathcal{C}$ une classe vérifiant les propriétés de l'énoncé du théorème. Soient $\mathcal{S} = (X, \rightarrow, \leq) \in \mathcal{C}$ et $x \in X$. Nous adaptons la stratégie de [FPS01] en

construisant un arbre d'accessibilité fini T avec racine r étiquetée par x. Si $\mathsf{Succ}(x)$ est infini, nous retournons « infini ». Autrement, nous marquons r et nous ajoutons un enfant y à r pour tout $y \in \mathsf{Succ}(x)$. Nous poursuivons ensuite la construction de l'arbre de la façon suivante. Un sommet non marqué s étiqueté y est choisi et,

- si s possède un ancêtre s' étiqueté y' tel que $y' < y$, nous retournons « infini » ;
- sinon, si s possède un ancêtre s' étiqueté y' tel que $y' = y$, nous marquons s ;
- sinon, si $\mathsf{Succ}(y)$ est infini, nous retournons « infini » ;
- sinon, nous marquons s, et pour tout $z \in \mathsf{Succ}(y)$ nous ajoutons à s un enfant étiqueté par z.

Nous montrons d'abord que cette procédure termine toujours. Notons d'abord que T est à branchement fini puisque des enfants sont ajoutés à un sommet étiqueté par y seulement lorsque $\mathsf{Succ}(y)$ est fini. Afin d'obtenir une contradiction, supposons que T est infini, alors, par le lemme de König, T possède une branche infinie étiquetée par certains $y_1, y_2, \ldots \in X$. Puisque $\leq$ est un beau préordre, il existe $i < j$ tels que $y_i \leq y_j$. Si $y_i < y_j$, alors la procédure aurait dû s'arrêter et retourner « infini ». Si $y_i = y_j$, alors le sommet étiqueté par y_j aurait dû être marqué. Ainsi, il n'existe pas de branche infinie, ce qui est une contradiction.

Nous prouvons maintenant que l'algorithme est correct. Nous remarquons que puisque $\mathcal{S}$ possède la monotonicité stricte et que $\leq$ est un bel ordre, $\mathsf{Succ}^*(x)$ est infini si, et seulement si, il existe une exécution[1] $x = y_0 \to y_1 \to \ldots \to y_k$ telle que

(a) $y_i \neq y_j$ pour tout $i \neq j$, et

(b) $\mathsf{Succ}(y_k)$ est infini ou $y_m < y_k$ pour un certain $m < k$.

Ainsi, lorsque l'algorithme retourne « infini », $\mathsf{Succ}(x)$ est bien infini.

De façon réciproque, supposons qu'il existe une exécution $x = y_0 \to y_1 \to \ldots \to y_k$ qui satisfait les points (a) et (b), alors elle apparait forcément dans T. Si

[1] Notons que cette équivalence n'est pas nécessairement vraie si $\leq$ est un beau préordre plutôt qu'un bel ordre, puisque $x \neq y$ et $x \leq y$ n'impliquent pas nécessairement $x < y$ sans antisymétrie.

$\mathsf{Succ}(y_k)$ est infini, l'algorithme retourne correctement « infini ». Sinon, l'algorithme retourne également « infini » puisque $y_m < y_k$. □

3.3.3 Problème de maintenabilité et de maintenabilité faible

Rappelons que le problème de maintenabilité consiste à déterminer s'il existe, à partir d'un état initial x, une exécution maximale qui demeure dans un certain ensemble clos par le haut. Il est connu que dans le cas à branchement fini, ce problème est décidable grâce aux propriétés de la monotonicité bégayante :

Théorème 3.35 ([FPS01])**.** *Le problème de maintenabilité est décidable pour toute classe post-effective $\mathcal{C}$ de WSTS à branchement fini avec monotonicité bégayante.*

Contrairement au problème de terminaison, le problème de maintenabilité et le problème de maintenabilité faible ne coïncident pas exactement dans le cas à branchement fini. Par exemple, considérons $\mathcal{S} = (\mathbb{N}, \rightarrow, \leq)$ le WSTS tel que $\rightarrow \stackrel{\text{déf}}{=} \emptyset$. Posons, par exemple, $x = 0$ et $A = \uparrow 0$. Nous avons $\mathsf{ExecMax}_A(x) \neq \emptyset$ puisque l'exécution triviale $x \rightarrow^0 x$ est maximale. Ainsi, x est « maintenable » dans A à partir de x. Cependant, x n'est pas « maintenanble faiblement », puisqu'il n'existe qu'une seule exécution à partir de x, et en particulier aucune exécution de longueur plus grande ou égale à deux.

Il existe toutefois une relation similaire à celle présentée à la proposition 3.28 dont la preuve en est calquée :

Proposition 3.36. *Soient $\mathcal{S} = (X, \rightarrow)$ un système de transitions, $x \in X$ et $A \subseteq X$. Nous avons $\mathsf{Exec}_A^{\mathit{inf}}(x) \neq \emptyset \iff \forall m \in \mathbb{N}, \exists y \in \mathsf{Exec}_A(x)$ telle que $|y| \geq m$.*

Cette observation permet d'adapter la preuve de [FPS01] afin d'obtenir le même résultat de décidabilité pour le problème de maintenabilité faible.

Théorème 3.37. *Le problème de maintenabilité faible est décidable pour toute classe post-effective $\mathcal{C}$ de WSTS à branchement fini avec monotonicité bégayante.*

Démonstration. Soit $\mathcal{C}$ une classe vérifiant les propriétés de l'énoncé du théorème. Soient $\mathcal{S} = (X, \rightarrow, \leq) \in \mathcal{C}$, $x \in X$ et $a_1, a_2, \ldots, a_n \in X$. Posons $A =$

$\uparrow\{a_1, a_2, \ldots, a_n\}$. Par la proposition 3.36 et l'exemple qui la précède, nous remarquons que ce qui distingue le problème de maintenabilité du problème de maintenabilité faible est l'éventuelle existence d'exécutions maximales finies.

Ainsi, nous adoptons l'algorithme présenté par [FPS01] afin de ne pas tenir compte des exécutions maximales finies. Plus précisément, nous construisons l'arbre d'accessibilité fini T des successeurs de x en élaguant chaque sommet qui possède un ancêtre plus petit, c.-à-d. nous n'ajoutons pas d'enfants à un sommet étiqueté par y qui possède un ancêtre étiqueté par $y' \leq y$. En suivant [FPS01], il est possible de montrer que, par monotonicité bégayante, $\mathsf{Exec}_A^{\mathsf{inf}}(x) \neq \emptyset$ si, et seulement si, il existe une branche débutant à la racine de T telle que $x = z_0 \rightarrow_A z_1 \rightarrow_A \ldots \rightarrow_A z_k$ et $z_i \leq z_k$ pour un certain $0 \leq i < k$. La seule modification de l'algorithme de [FPS01] est donc la condition que « $z_i \leq z_k$ » pour s'assurer de l'existence d'une exécution non seulement maximale, mais également infinie. □

Nous remarquons que sans la monotonicité bégayante, le problème de maintenabilité et le problème de maintenabilité faible deviennent tous deux indécidables :

Théorème 3.38. *Il existe une classe post-effective de WSTS-ω^2 à branchement fini avec bel ordre pour laquelle le problème de maintenabilité et le problème de maintenabilité faible sont indécidables.*

Démonstration. Soit $\mathcal{C}(i) \stackrel{\text{déf}}{=} (\mathbb{N}, \rightarrow, \leq)$ le système de transitions ordonné défini par

- $x \rightarrow x + 1$ si $x \geq 1$ et Turing_i ne s'arrête pas sur son encodage en x étapes ou moins,
- $x \rightarrow 0$ si $x \geq 1$ et Turing_i s'arrête sur son encodage en x étapes ou moins.

Nous remarquons que $\mathcal{C}$ est une classe qui vérifie les propriétés de l'énoncé du théorème.

Posons $x = 1$ et $A = \uparrow 1$. Notons d'abord que $\mathsf{ExecMax}_A^{\mathsf{fin}}(x) = \emptyset$. En effet, supposons au contraire qu'il existe une exécution finie $y \in \mathsf{ExecMax}_A^{\mathsf{fin}}(x)$, alors

$\mathsf{Succ}(y_{|y|}) = \emptyset$, et ainsi $y_{|y|} = 0$ par définition de $\rightarrow$. Or, $y_{|y|} \in A$, ce qui est une contradiction.

Par cette observation et par la proposition 3.36, nous avons donc

$$\begin{aligned} \mathsf{ExecMax}_A(x) \neq \emptyset &\iff \mathsf{Exec}^{\mathsf{inf}}_A(x) \neq \emptyset \\ &\iff \forall m \in \mathbb{N}, \exists y \in \mathsf{Exec}_A(x) \text{ t.q. } |y| \geq m \ . \end{aligned}$$

Ainsi, le problème de maintenabilité et le problème de maintenabilité faible sont identiques pour $\mathcal{C}$. De plus, pour tout i, $\mathsf{Exec}^{\mathsf{inf}}_A(x) \neq \emptyset$ si, et seulement si, la machine Turing_i ne s'arrête pas sur son encodage. Ainsi, les deux problèmes sont indécidables. □

Nous étudions maintenant le cas des WSTS à branchement infini. Nous donnons d'abord un second exemple où le problème de maintenabilité diffère du problème de maintenabilité faible. Cet exemple moins artificiel que celui présenté plus tôt montre que lorsque le branchement infini est autorisé, la distinction entre les deux problèmes s'accentue. En effet, reconsidérons le réseau Petri-ω introduit à la section 3.3.1 et illustré à la figure 3.2, mais maintenant avec le marquage initial $\boldsymbol{x} = (1, 1, 0)$. Posons $A = \uparrow(0, 1, 0)$. Tel qu'illustré à la figure 3.4, il n'existe pas d'exécution maximale qui demeure dans A, ainsi ce réseau de Petri-ω n'est pas « maintenable » dans A à partir de x. Or, il est « maintenable faiblement » puisque la longueur des exécutions demeurant dans A à partir de x n'est pas bornée.

Nous montrons que contrairement au cas à branchement fini, le problème de maintenabilité est indécidable sous les hypothèses les plus fortes introduites dans cette thèse :

Théorème 3.39. *Il existe une classe, post-effective et post-effective sous complétion, de WSTS-ω^2 avec monotonicité forte et stricte, et bel ordre, pour laquelle le problème de maintenabilité est indécidable.*

Démonstration. Nous donnons une réduction à partir du problème de (non) terminaison qui a été démontré indécidable au théorème 3.31 pour une classe $\mathcal{C}$ respectant

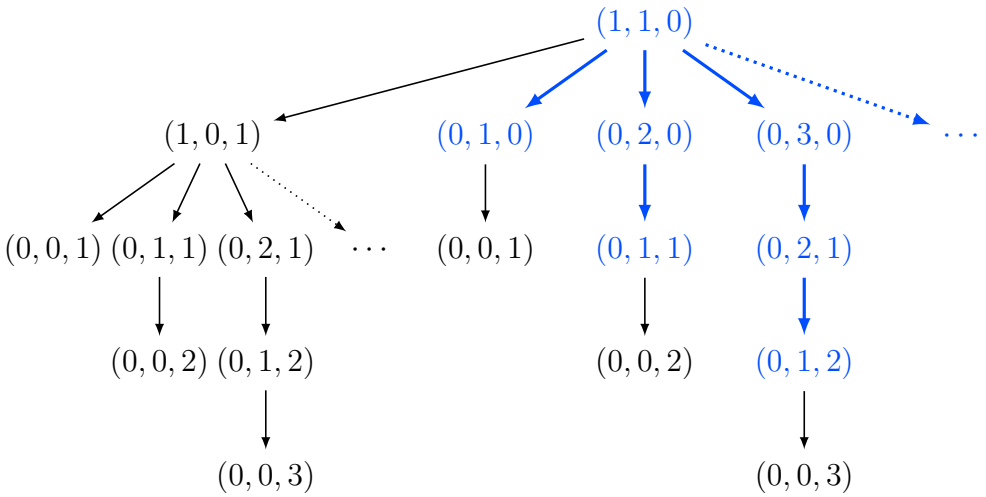


Figure 3.4 – Arbre d'accessibilité, à partir de $\boldsymbol{x} = (1, 1, 0)$, du réseau de Petri-$\omega$ illustré à la figure 3.2. Les exécutions demeurant dans $A = \uparrow(0, 1, 0)$ sont identifiées en couleur ( ). Aucune de ces exécutions n'est maximale.

les mêmes propriétés.

Soient $\mathcal{C}(i) = (X, \rightarrow, \leq)$ et $x \in X$. Soient $(X', \rightarrow', \leq')$ une copie disjointe de $\mathcal{C}(i)$, et $\mathcal{D}(i) \stackrel{\text{déf}}{=} (X \cup X' \cup \{x_{\min}\}, \rightarrow'', \leq'')$ le système de transitions ordonné défini par

- $\rightarrow'' \stackrel{\text{déf}}{=} \rightarrow \cup \rightarrow' \cup \{(x, x') : x \in X)\}$,
- $\leq'' \stackrel{\text{déf}}{=} \leq \cup \leq' \cup \{(x_{\min}, x) : x \in X \cup \{x_{\min}\}\}$.

Il peut être démontré que $\mathcal{D}(i)$ est un WSTS-ω^2 avec monotonicité forte et stricte, et que $\leq''$ est un bel ordre.

De plus, $\mathcal{D}$ est post-effective et post-effective sous complétion. En effet, la copie disjointe de X peut être encodée en utilisant un nouveau symbole servant de préfixe à l'encodage original de X ; $x_{\min}$ peut également être encodé par un symbole additionnel. L'encodage des idéaux de $X \cup X' \cup \{x_{\min}\}$, c.-à-d., $\{I \cup \{x_{\min}\} : I \in \mathsf{Ideaux}(X)\}$ et une copie disjointe de $\mathsf{Ideaux}(X)$, peut aisément être adapté de l'encodage original de $\mathsf{Ideaux}(X)$. Les machines de Turing calculant $\rightarrow$, $\leq$, $\rightsquigarrow$ et $\subseteq$ peuvent aussi facilement être adaptées à ces nouveaux encodages et relations.

Posons $A = \uparrow x_{\min}$. Montrons que $\mathsf{Exec}^{\mathsf{inf}}(x) \neq \emptyset$ dans $\mathcal{C}(i)$ si, et seulement si, $\mathsf{ExecMax}_A(x) \neq \emptyset$ dans $\mathcal{D}(i)$. Supposons qu'il existe une exécution infinie $x = y_0 \to y_1 \to \dots$ dans $\mathcal{C}(i)$. Cette séquence est aussi une exécution dans $\mathcal{D}(i)$. De plus, $x_{\min} \leq'' y_i$ pour tout $i \in \mathbb{N}$. Ainsi, $x = y_0 \to''_A y_1 \to''_A \dots$ dans $\mathcal{D}(i)$.

Afin de montrer la réciproque, supposons que $\mathsf{ExecMax}_A(x) \neq \emptyset$ dans $\mathcal{D}(i)$. Montrons d'abord qu'il n'existe pas d'exécution maximale finie dans $\mathcal{D}(i)$. Supposons qu'il en existe une, alors il existe $y \in \mathsf{ExecMax}^{\mathsf{fin}}_A(x)$. Nous avons $y_j \in X$ pour tout $1 \leq j \leq |y|$, puisque $X' \cap A = \emptyset$ et que $x_{\min}$ est inaccessible à partir de x. Ainsi, y peut être étendue grâce à $y'_{|y|} \in \mathsf{Succ}_{\mathcal{D}(i)}(y_{|y|})$ ce qui est une contradiction puisque y est maximale. Ainsi, par hypothèse, il existe $y \in \mathsf{Exec}^{\mathsf{inf}}_A(x)$ dans $\mathcal{D}(i)$. À nouveau, nous avons $y_j \in X$ pour tout $j \in \mathbb{N}_{>0}$. Puisque chaque état est dans X, la même exécution infinie existe dans $\mathcal{D}(i)$, ce qui termine la preuve. □

En revanche, nous montrons que le problème de maintenabilité faible est décidable pour les classes de WSTS avec monotonicité forte pourvu que leur complétion satisfasse quelques propriétés additionnelles :

Théorème 3.40. *Le problème de maintenabilité faible est décidable pour toute classe, post-effective sous complétion, de WSTS-ω^2 avec monotonicité forte.*

Notre preuve du théorème 3.40 nécessite les propositions 3.41 et 3.42, qui relient les exécutions, demeurant dans des ensembles clos par le haut, d'un WSTS à celles de sa complétion.

Proposition 3.41. *Soient $\mathcal{S} = (X, \to, \leq)$ un WSTS avec monotonicité forte, et $a_1, a_2, \dots, a_n \in X$. Posons $U = \uparrow_{\mathsf{Ideaux}(X)} \{\downarrow a_1, \downarrow a_2, \dots, \downarrow a_m\}$ et $A = \uparrow\{a_1, a_2, \dots, a_n\}$. Si $I_0 \rightsquigarrow_U I_1 \rightsquigarrow_U \dots \rightsquigarrow_U I_k$, alors pour tout $y \in I_k$ il existe une exécution $x_0 \to_A x_1 \to_A \dots \to_A x_k$ telle que $x_0 \in I_0$ et $x_k \geq y$.*

Démonstration. Nous procédons par induction sur k. Supposons que $k = 0$. Soit $y \in I_0$. Par définition de U, il existe $i \in [n]$ tel que $\downarrow a_i \subseteq I_0$ et ainsi $a_i \in I_0$. Puisque I_0 est un idéal, il existe $x_0 \in I_0$ tel que $y \leq x_0$ et $a_i \leq x_0$. Ainsi, $x_0 \in A$, $x_0 \in I_0$ et $x_0 \geq y$.

Supposons que $k > 0$ et que $I_0 \rightsquigarrow_U I_1 \rightsquigarrow_U \ldots \rightsquigarrow_U I_k$. Par hypothèse d'induction, pour tout $y \in I_k$, il existe une exécution $x_1 \rightarrow_A x_2 \rightarrow_A \ldots \rightarrow_A x_k$ telle que $x_1 \in I_1$ et $x_k \geq y$. Puisque $x_1 \in I_1 \subseteq {\downarrow}\mathsf{Succ}_{\mathcal{S}}(I_0)$, il existe $x_0 \in I_0$ et $y' \geq x_1$ tels que $x_0 \rightarrow y'$. Par définition de U, il existe $i \in [n]$ tel que ${\downarrow}a_i \subseteq I_0$ et ainsi $a_i \in I_0$. Puisque I_0 est un idéal, il existe $x'_0 \in I_0$ tel que $x_0 \leq x'_0$ et $a_i \leq x'_0$. Par monotonicité forte, il existe $x'_1 \geq y'$ tel que $x'_0 \rightarrow x'_1$. Par définition de A, ${\downarrow}a_j \subseteq I_1$ pour un certain $j \in [n]$. Ainsi, $x'_0 \rightarrow_A x'_1$ puisque $x'_1 \geq y' \geq x_1$ et $x_1 \in A$. De plus, en appliquant la monotonicité forte à $x_1 \rightarrow_A x_2 \rightarrow_A \ldots \rightarrow_A x_k$ avec $x'_1 \geq x_1$, nous obtenons une exécution $x'_1 \rightarrow_A x'_2 \rightarrow_A \ldots \rightarrow_A x'_k$ telle que $x'_j \geq x_j$ pour tout $j \in [k]$. Ainsi, $x'_0 \rightarrow_A x'_1 \rightarrow_A \ldots \rightarrow_A x'_k$, $x'_0 \in I_0$ et $x'_k \geq y$. □

Proposition 3.42. *Soient $\mathcal{S} = (X, \rightarrow, \leq)$ un WSTS et $a_1, a_2, \ldots, a_n \in X$. Posons $A = {\uparrow}\{a_1, a_2, \ldots, a_n\}$ et $U = {\uparrow}_{\mathsf{Ideaux}(X)}\{{\downarrow}a_1, {\downarrow}a_2, \ldots, {\downarrow}a_n\}$. Si $x_0 \rightarrow_A x_1 \rightarrow_A \ldots \rightarrow_A x_k$, alors pour tout idéal $I_0 \supseteq {\downarrow}x_0$ il existe une exécution $I_0 \rightsquigarrow_U I_1 \rightsquigarrow_U \ldots \rightsquigarrow_U I_k$ telle que $I_k \supseteq {\downarrow}x_k$.*

Démonstration. Nous procédons par induction sur k. Supposons que $k = 0$. Soit I_0 un idéal tel que $I_0 \supseteq {\downarrow}x_0$. Par définition de A, il existe $i \in [n]$ tel que $x_0 \geq a_i$. Ainsi, $I_0 \supseteq {\downarrow}x_0 \supseteq {\downarrow}a_i$, et par conséquent $I_0 \in U$.

Supposons que $k > 0$ et $x_0 \rightarrow_A x_1 \rightarrow_A \ldots \rightarrow_A x_k$. Soit I_0 un idéal tel que $I_0 \supseteq {\downarrow}x_0$. Nous avons $x_1 \in \mathsf{Succ}_{\mathcal{S}}(I_0) \subseteq {\downarrow}\mathsf{Succ}_{\mathcal{S}}(I_0)$. Ainsi, $I_0 \rightsquigarrow I_1$ pour un certain idéal $I_1 \supseteq \{x_1\}$. Puisque I_1 est clos par le bas, nous avons $I_1 \supseteq {\downarrow}x_1$. De plus, par définition de A, il existe $i \in [n]$ tel que $x_0 \geq a_i$. Ainsi, $I_0 \supseteq {\downarrow}x_0 \supseteq {\downarrow}a_i$ et nous avons $I_0 \in U$. Par hypothèse d'induction, il existe une exécution $I_1 \rightsquigarrow_U I_2 \rightsquigarrow_U \ldots \rightsquigarrow_U I_k$ telle que $I_k \supseteq {\uparrow}x_k$. Nous concluons donc que $I_0 \rightsquigarrow_U I_1 \rightsquigarrow_U \ldots \rightsquigarrow_U I_k$ et $I_k \supseteq {\downarrow}x_k$. □

Nous sommes maintenant en mesure de prouver le théorème 3.40 à partir des propositions 3.41 et 3.42.

Preuve du théorème 3.40. Soit $\mathcal{C}$ une classe telle que décrite dans l'énoncé de la proposition. Soit $\mathcal{S} \in \mathcal{C}$, puisque $\mathcal{S}$ est un WSTS-ω^2, $\widehat{\mathcal{S}}$ est un WSTS. De plus,

$\widehat{\mathcal{S}}$ possède la monotonicité forte par la proposition 3.18 et ainsi la monotonicité bégayante. Puisque $\mathcal{C}$ est effective sous complétion, $\widehat{\mathcal{C}}$ est post-effective. Ainsi, $\widehat{\mathcal{C}}$ est une classe post-effective de WSTS à branchement fini avec monotonicité bégayante. Par le théorème 3.37, le problème de maintenabilité faible est donc décidable pour $\widehat{\mathcal{C}}$.

Montrons que cela implique que le problème est aussi décidable pour $\mathcal{C}$. Soient $\mathcal{S} = (X, \to, \leq) \in \mathcal{C}$, $x \in X$ et $a_1, a_2, \ldots, a_n \in X$. Posons $A = \uparrow\{a_1, a_2, \ldots, a_n\}$ et $U = \uparrow_{\mathsf{Ideaux}(X)} \{\downarrow a_1, \downarrow a_2, \ldots, \downarrow a_n\}$. Nous prétendons qu'il n'existe pas de borne sur la longueur des exécutions demeurant dans A à partir de x dans $\mathcal{S}$ si, et seulement si, il n'existe pas de borne sur la longueur des exécutions demeurant dans U à partir de $\downarrow x$ dans $\widehat{\mathcal{S}}$. Ainsi, il est possible de décider la maintenabilité faible à partir de x dans $\mathcal{S}$ en la décidant plutôt à partir de $\downarrow x$ dans $\widehat{\mathcal{S}}$. Puisque $\mathcal{C}$ est post-effective sous complétion, cela est possible en obtenant l'encodage de $\downarrow x$ à partir de x.

Supposons qu'il n'existe pas de borne $m \in \mathbb{N}$ sur la longueur des exécutions de $\mathsf{Exec}_A(x)$ dans $\mathcal{S}$. Afin d'obtenir une contradiction, supposons qu'il existe une borne $m \in \mathbb{N}$ sur la longueur des exécutions de $\mathsf{Exec}_U(\downarrow x)$ dans $\widehat{\mathcal{S}}$. Soit $y \in \mathsf{Exec}_A(x)$ dans $\mathcal{S}$ telle que $|y| = m' > m$. Par la proposition 3.42, il existe une exécution $\downarrow x = I_0 \rightsquigarrow_U I_1 \rightsquigarrow_U \cdots \rightsquigarrow_U I_{m'}$, ce qui est une contradiction. Ainsi, la longueur des exécutions de $\mathsf{Exec}_U(\downarrow x)$ dans $\widehat{\mathcal{S}}$ n'est pas bornée.

De façon réciproque, supposons qu'il n'existe pas de borne $m \in \mathbb{N}$ sur la longueur des exécutions de $\mathsf{Exec}_U(\downarrow x)$ dans $\widehat{\mathcal{S}}$. Afin d'obtenir une contradiction, supposons qu'il existe une borne $m \in \mathbb{N}$ sur la longueur des exécutions de $\mathsf{Exec}_A(x)$ dans S. Soit $J \in \mathsf{Exec}_U(\downarrow x)$ dans $\widehat{\mathcal{S}}$ telle que $|J| = m' > m$. Par la proposition 3.41, il existe une exécution $x_0 \to_A x_1 \to_A \cdots \to_A x_{m'}$ telle que $x_0 \in \downarrow x$. Par monotonicité forte de $\mathcal{S}$, il existe une exécution $x = x'_0 \to_A x'_1 \to_A \cdots \to_A x'_{m'}$ où $x'_i \geq x_i$ pour tout $i \in [m']$. Cela est une contradiction. □

3.3.4 Problème de couverture

Nous terminons notre étude des problèmes de décision par le problème de couverture. Rappelons que le problème de couverture consiste à déterminer si un état

y est couvrable à partir d'un état x, c.-à-d., si $x \rightarrow^* y'$ pour un certain $y' \geq y$, ou de façon équivalente si $y \in \downarrow \mathsf{Succ}^*(x)$. Nous notons d'abord que le problème de couverture est indécidable sous plusieurs hypothèses :

Théorème 3.43 ([FMP04])**.** *Il existe une classe post-effective de WSTS-ω^2 à branchement fini avec monotonicité forte et stricte, et bel ordre, pour laquelle le problème de couverture est indécidable.*

Le problème de couverture est néanmoins décidable pour certaines classes. Contrairement aux autres problèmes étudiés dans ce chapitre, les algorithmes existants [FPS01, ACJT96], afin de résoudre le problème de couverture, fonctionnent pour certaines classes de WSTS à branchement infini. Ces algorithmes procèdent typiquement par *méthode arrière* ; afin de déterminer si y est couvrable à partir de x, plutôt que de vérifier si $y \in \downarrow \mathsf{Succ}^*(x)$, ils vérifient, de façon équivalente, si $x \in \mathsf{Pred}^*(\uparrow y)$. La raison pour laquelle ces algorithmes fonctionnent ainsi est motivée par le fait que les WSTS sont munis de beaux préordres, et qu'ainsi la séquence d'ensembles clos par le haut

$$\uparrow y,\ \uparrow \mathsf{Pred}(\uparrow y),\ \uparrow \mathsf{Pred}(\uparrow \mathsf{Pred} \uparrow y)),\ \ldots \tag{3.3}$$

se stabilise en un nombre fini d'étapes (voir section 2.4.2). De plus l'union finie de ces ensembles est égale à $\uparrow \mathsf{Pred}^*(\uparrow y) = \mathsf{Pred}^*(\uparrow y)$ (p. ex., voir [FPS01, section 3]). Ainsi, plutôt que d'effectuer un calcul des successeurs qui ne se stabilise pas nécessairement [FPS01], la méthode arrière exploite le calcul arrière qui se stabilise toujours.

Tel qu'illustré à l'algorithme 3.1, le $i^{\text{ème}}$ ensemble de la séquence (3.3) peut être représenté par sa base finie minimale B_i. De plus, le calcul de $\mathsf{MinBase}$[2] est simple à calculer, tel que mentionné à la section 2.4.2. Ainsi, l'implémentation de l'algo-

[2]Notons que l'algorithme 3.1 fonctionne correctement même en ne minimisant pas les bases.

Algorithme 3.1 : Algorithme arrière pour le problème de couverture pour les WSTS.

Entrées : $\mathcal{S} = (X, \to, \leq) \in \mathcal{C}$ et $x, y \in X$
Sorties : $\exists y' \geq y$ tel que $x \to y'$?

$i \leftarrow 0$
$B_0 \leftarrow \{y\}$
répéter
    $C \leftarrow \mathsf{MinBase}(\uparrow \mathsf{Pred}(\uparrow B_i))$
    $B_{i+1} \leftarrow \mathsf{MinBase}(B_i \cup C)$
    $i \leftarrow i + 1$
jusqu'à $B_i = B_{i-1}$
retourner $x \in \uparrow B_i$

rithme 3.1 dépend du fait qu'il soit possible de calculer une base finie de

$$\uparrow \mathsf{Pred}(\uparrow B_i) = \bigcup_{b \in B_i} \uparrow \mathsf{Pred}(\uparrow b) \ .$$

Nous définissons formellement cette hypothèse d'effectivité :

Definition 3.44 ([FPS01, ACJT96])**.** Une classe $\mathcal{C}$ de systèmes de transitions ordonnés est dite *pré-effective sous clôture par le haut*[3] s'il existe une machine de Turing M_{prebase} qui calcule une base minimale de $\uparrow \mathsf{Pred}_{\mathcal{C}(i)}(\uparrow x)$ sur entrée (i, x), où $i \in \mathbb{N}$ et x est un état de $\mathcal{C}(i)$. Par extension, nous disons qu'un WSTS $\mathcal{S}$ est *pré-effectif sous clôture par le haut* si la classe dégénérée $\{\mathcal{S}\}$ est pré-effective sous clôture par le haut.

Nous obtenons donc formellement le résultat suivant de décidabilité :

Théorème 3.45 ([FPS01, ACJT96])**.** *Le problème de couverture est décidable pour toute classe effective $\mathcal{C}$ de WSTS pré-effective sous clôture par le haut.*

À l'instar de la post-effectivité, sous complétion ou non, les classes de WSTS ne sont pas toutes pré-effectives sous clôture par le haut. Par exemple, les processus de profondeur bornée [ZWH12] ne sont pas pré-effectifs sous clôture par le haut.

[3] Nous choisissons cette terminologie plutôt que « pré-effective » puisque celle-ci correspondrait plutôt au simple calcul de $\mathsf{Pred}(x)$.

De plus, la pré-effectivité sous clôture par le haut n'est pas nécessaire à la décidabilité du problème de couverture. En effet, le problème de couverture est notamment décidable pour la classe des processus de profondeur bornée [ZWH12] grâce à l'algorithme EEC, de l'anglais « Expand, Enlarge and Check », de [GRB06a], partiellement reformulé dans le cadre de la complétion d'idéaux de [FG09]. Notons également qu'il existe des approches qui n'exploitent pas la méthode arrière, par exemple, Geeraerts *et al.* utilisent un algorithme *avant* [GRB06b] qui requiert des hypothèses algébriques.

Nous montrons que le problème de couverture est décidable sous une hypothèse d'effectivité alternative, c'est-à-dire, pour les classes post-effectives sous complétion. Notre approche est donc basée sur un calcul avant plutôt qu'arrière.

Théorème 3.46. *Le problème de couverture est décidable pour toute classe de WSTS post-effective sous complétion.*

Démonstration. Soit $\mathcal{C}$ une classe de $WSTS$ post-effective sous complétion. Soient $S = (X, \to, \leq) \in \mathcal{C}$, et $x, y \in X$. Nous montrons comment décider si y est couvrable à partir de x à l'aide de deux « semi-procédures ». Autrement dit, nous cherchons itérativement, et en alternance, à prouver d'un côté que y est couvrable, et de l'autre à prouver que y n'est pas couvrable, jusqu'à l'obtention d'une réponse.

Montrons que y est couvrable à partir de x dans $\mathcal{S}$ si, et seulement si, $\downarrow y$ est couvrable à partir de $\downarrow x$ dans $\widehat{\mathcal{S}}$. Si y est couvrable, alors il existe $x \to^* y$. Ainsi, par la proposition 3.16, il existe un idéal $J \supseteq \downarrow y$ tel que $\downarrow x \rightsquigarrow^* J$, et par conséquent $\downarrow y$ est couvrable. Supposons que $\downarrow y$ soit couvrable, alors il existe un idéal $J \supseteq \downarrow y$ tel que $\downarrow x \rightsquigarrow^* J$. Ainsi, par la proposition 3.15, il existe $x' \in \downarrow x$ et $y' \geq y$ tels que $x' \to^* y'$. Par monotonicité, $x \to^* y''$ pour un certain $y'' \geq y'$. Puisque $y'' \geq y' \geq y$, y est couvrable.

Ainsi, afin de tester si y est couvrable, nous construisons itérativement l'arbre d'accessibilité à branchement fini de $\widehat{\mathcal{S}}$ à partir de $\downarrow x$. Autrement dit, nous calculons successivement les ensembles finis $\downarrow x$, $\mathsf{Succ}_{\widehat{\mathcal{S}}}(\downarrow x)$, $\mathsf{Succ}_{\widehat{\mathcal{S}}}(\mathsf{Succ}_{\widehat{\mathcal{S}}}(\downarrow x))$, ... et nous testons si l'un d'eux contient un idéal J tel que $J \supseteq \downarrow y$. Ces calculs sont rendus

possibles par le fait que $\widehat{\mathcal{S}}$ est effective.

D'autre part, nous testons si y n'est pas couvrable. Remarquons que y n'est pas couvrable à partir de x si, et seulement si, il existe un invariant inductif D tel que $x \in D$ et $y \notin D$. Un *invariant inductif* est un ensemble clos par le bas $D \subseteq X$ tel que $\downarrow \mathsf{Succ}_{\mathcal{S}}(D) \subseteq D$. Par une induction simple, nous observons qu'un invariant inductif satisfait la propriété $\downarrow \mathsf{Succ}^*_{\mathcal{S}}(D) \subseteq D$. Ainsi, si $x \in D$ et $y \notin D$, cela démontre que y n'est pas couvrable. Pour s'en convaincre, supposons au contraire que y soit couvrable. Alors, $y \in \downarrow \mathsf{Succ}^*_{\mathcal{S}}(x) \subseteq \downarrow \mathsf{Succ}^*_{\mathcal{S}}(D) \subseteq D$, ce qui est une contradiction. Afin de prouver que y n'est pas couvrable, il suffit donc d'énumérer les invariants inductifs D et de tester si $x \in D$ et $y \notin D$. Notons que cela est possible par post-effectivité de $\widehat{\mathcal{S}}$. Les ensembles D clos par le bas peuvent être énumérés par leur décomposition en idéaux. Afin de tester si D est un invariant inductif, nous exploitons l'observation suivante où la dernière équivalence découle de la proposition 3.7 :

$$\downarrow \mathsf{Succ}_{\mathcal{S}}(D) \subseteq D \iff \bigcup_{I \in \mathsf{DecompIdeaux}(D)} \downarrow \mathsf{Succ}_{\mathcal{S}}(I) \subseteq \bigcup_{I \in \mathsf{DecompIdeaux}(D)} I$$

$$\iff \bigcup_{I \in \mathsf{DecompIdeaux}(D)} \bigcup_{J \in \mathsf{Succ}_{\widehat{\mathcal{S}}}(I)} J \subseteq \bigcup_{I \in \mathsf{DecompIdeaux}(D)} I$$

$$\iff \forall J \in \mathsf{Succ}_{\widehat{\mathcal{S}}}(\mathsf{DecompIdeaux}(D)),$$
$$\exists I \in \mathsf{DecompIdeaux}(D) \text{ t.q. } J \subseteq I \ . \quad (3.4)$$

Puisque $\widehat{\mathcal{S}}$ est post-effective, il est possible de calculer $\mathsf{Succ}_{\widehat{\mathcal{S}}}$ et de tester l'inclusion entre idéaux, et ainsi d'effectuer les calculs en (3.4). □

Notons que la technique qui consiste à énumérer des invariants inductifs, utilisée dans notre algorithme pour le problème de couverture, fut utilisée par Pachl afin de fournir des témoins de non accessibilité pour certaines classes d'automates finis communiquant à l'aide de files [Pac82, corollaire 9.6]. Plus récemment, Raskin et

al. [GRB06a, GRB06b] ont utilisé des techniques d'énumération similaires afin de mettre au point des algorithmes avant décidant la couverture pour les WSTS. Cependant, leurs techniques utilisent des hypothèses algébriques et d'effectivité additionnelles.

Nous remarquons que bien que certaines classes peuvent à la fois être pré-effectives sous clôture par le haut, et post-effectives sous complétion, notre approche effectue un calcul avant plutôt qu'arrière. En pratique, il est souvent plus efficace de calculer Succ que Pred. En revanche, le nombre d'itérations peut être élevé. À titre d'exemple, nous montrons que pour les réseaux affines, le calcul d'une base minimale de $\uparrow\mathsf{Pred}(\uparrow x)$ est NP-difficile alors qu'il est possible de calculer $\mathsf{DecompIdeaux}(\downarrow\mathsf{Succ}(x))$ en temps polynomial tel que mentionné à l'exemple 3.26 :

Proposition 3.47. *Le problème suivant est* NP*-difficile sous réduction Turing log-espace.*

Entrée: *Un réseau affine* $\mathcal{S}$ *à* d *compteurs,* $\boldsymbol{y} \in \mathbb{N}^d$*,* $i, j \in \mathbb{N}$ *et* $b \in \{0, 1\}$*.*

Question: *Le* $i^{\text{ème}}$ *bit du* $j^{\text{ème}}$ *vecteur, en ordre de poids*[4]*, de* $\mathsf{MinBase}(\uparrow \mathsf{Pred}_{\mathcal{S}}(\uparrow y))$ *est-il égal à* b *?*

Avant de prouver la proposition 3.47, nous devons démontrer qu'une variante de problème de programmation linéaire entière est NP-complet.

Proposition 3.48. *Le problème suivant est* NP*-complet sous réduction multivoque log-espace.*

Entrée: $\mathbf{A} \in \{0, 1\}^{m \times n}$*,* $k \in \mathbb{N}$ *encodés en unaire.*

Question: *Existe-t-il* $\boldsymbol{x} \in \mathbb{N}^n$ *tel que* $\mathbf{A}\boldsymbol{x} \geq \mathbf{1}$ *et* $\sum_{i=1}^{n} \boldsymbol{x}(i) \leq k$ *?*

Démonstration. Nous donnons une réduction à partir de 3-SAT en nous inspirant de résultats similaires (p. ex., voir [GJ79, p. 245]). Soit la formule $\varphi(x_1, \ldots, x_u) = \bigwedge_{i \in [r]}(\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$ telle que chaque $\ell_{i,j}$ est un littéral. Nous construisons une matrice $\mathbf{A} \in \{0, 1\}^{(u+r)\times 2u}$ de telle sorte que $\boldsymbol{x}(2j-1)$ et $\boldsymbol{x}(2j)$ du vecteur solution $\boldsymbol{x}$ correspondent aux valeurs booléennes de la variable x_j pour une affectation qui

[4] Le poids d'un vecteur $\boldsymbol{x} \in \mathbb{N}^d$ est $\sum_{i=1}^{d} \boldsymbol{x}(i)$.

satisfait φ. Ainsi, nous construisons $\mathbf{A}$ de telle sorte que $\boldsymbol{x}(2j-1) = 1$ et $\boldsymbol{x}(2j) = 0$ indique que $x_j = \mathsf{vrai}$, et que $\boldsymbol{x}(2j-1) = 0$ et $\boldsymbol{x}(2j) = 1$ indique que $x_j = \mathsf{faux}$.

Afin de s'assurer que les vecteurs solutions soient binaires, nous ajoutons à $\mathbf{A}$, pour chaque $j \in [u]$, la contrainte

$$\boldsymbol{x}(2j-1) + \boldsymbol{x}(2j) \geq 1 \ . \tag{3.5}$$

Afin de s'assurer que les vecteurs solutions représentent les affectations qui satisfont φ, nous ajoutons à $\mathbf{A}$, pour chaque $i \in [r]$, la contrainte

$$\boldsymbol{x}(a_{i,1}) + \boldsymbol{x}(a_{i,2}) + \boldsymbol{x}(a_{i,3}) \geq 1 \tag{3.6}$$

où, pour tout $t \in \{1,2,3\}$, $a_{i,t} = 2j-1$ si $\ell_{i,t} = x_j$ et $a_{i,t} = 2j$ si $\ell_{i,t} = \neg x_j$. La matrice $\mathbf{A}$ est facilement construite en espace logarithmique à partir de φ.

Montrons que φ est satisfaisable si, et seulement si, il existe $\boldsymbol{x} \in \mathbb{N}^{2u}$ tel que $x \geq \mathbf{1}$ et $\sum_{i=1}^{2u} \boldsymbol{x}(i) \leq u$. Supposons que φ soit satisfaisable par une affectation $(y_1, y_2, \ldots, y_u)$, alors le système possède une solution $\boldsymbol{x}$, c.-à-d., le vecteur $\boldsymbol{x}$ tel que $\boldsymbol{x}(2j-1) = 1$ et $\boldsymbol{x}(2j) = 0$ si $y_j = \mathsf{vrai}$ et $\boldsymbol{x}(2j-1) = 0$ et $\boldsymbol{x}(2j) = 1$ si $y_j = \mathsf{faux}$. De façon réciproque, supposons qu'il existe une solution $\boldsymbol{x}$ au système. Nous avons $\sum_{i=1}^{2u} \boldsymbol{x}(i) = u$ et ainsi, par (3.5), $\boldsymbol{x}(2j-1) = 1 - \boldsymbol{x}(2j)$ pour tout $j \in [u]$. Il est donc possible d'extraire une affectation $(y_1, y_2, \ldots, y_u)$ de $\boldsymbol{x}$. Par (3.6), cette affectation satisfait φ. Cela montre que le problème est NP-difficile.

Nous concluons la preuve en remarquant que le problème est dans NP puisqu'il suffit de tester en temps polynomial si un vecteur $\boldsymbol{x} \in \{0, 1, \ldots, k\}^n$ est une solution. □

Nous sommes maintenant en mesure de prouver la proposition 3.47.

Preuve de la proposition 3.47. Nous donnons une réduction Turing à partir du problème NP-complet de la proposition 3.48. Supposons donc qu'il existe un oracle O pour notre problème. Soient $\mathbf{A} \in \mathbb{N}^{m \times n}$ et $k \in \mathbb{N}$ encodés en unaire. Soit $\mathcal{S} = (\{q\} \times \mathbb{N}^d, \rightarrow, \{f\}, \leq)$ le réseau affine tel que $f(q, \boldsymbol{x}) = (q, \mathbf{A}\boldsymbol{x} + \mathbf{0})$. Nous

remarquons que

$$\uparrow \mathsf{Pred}_{\mathcal{S}}(\uparrow \mathbf{1}) = \mathsf{Pred}_{\mathcal{S}}(\uparrow \mathbf{1}) = \{\boldsymbol{y} \in \mathbb{N}^d : \mathbf{A}\boldsymbol{y} \geq \mathbf{1}\} \ .$$

Nous faisons appel à l'oracle O afin d'obtenir le premier vecteur $\boldsymbol{x}_{\min}$, s'il en existe un, tel que $\boldsymbol{x}_{\min} \in \mathsf{MinBase}(\uparrow \mathsf{Pred}_{\mathcal{S}}(\uparrow \mathbf{1}))$. Par définition du problème, $\boldsymbol{x}_{\min}$ est de poids $\sum_{i=1}^{d} \boldsymbol{x}_{\min}(i)$ minimum. Notons que l'obtention de $\boldsymbol{x}_{\min}$ nécessite au plus d appels à l'oracle, puisque, par non négativité de $\mathbf{A}$, les vecteurs minimaux sont tous plus petits ou égaux à $\mathbf{1}$.

Ainsi, si $\boldsymbol{x}_{\min}$ existe et $\sum_{i=1}^{d} \boldsymbol{x}_{\min}(i) \leq k$, nous acceptons l'entrée, sinon nous refusons. □

Comme pour la pré-effectivité sous clôture par le haut, la post-effectivité sous complétion n'est pas nécessaire à la décidabilité du problème de couverture. Afin d'être complet, nous donnons un exemple de classe dont le problème de couverture est décidable, mais qui ne possède ni la pré-effectivité sous clôture par le haut, ni la post-effectivité sous complétion. Cette classe, que nous nommerons $\mathcal{F}_1$, est inspirée des réseaux bien stucturés de dimension 1 introduits dans [FMP04].

Nous définissons $\mathcal{F}_1$ de la façon suivante. Le $i^{\text{ème}}$ WSTS $\mathcal{S}$ de la classe, $\mathcal{S} = (\mathbb{N}, \to, \{f_1, f_2, \ldots, f_{n_i}\}, \leq)$, est un WSTS fonctionnel où, pour tout $j \in [n_i]$, f_j est une fonction qui dépend d'une machine de Turing $M_{i,j}$ de la façon suivante :

$$f_j(x) = \begin{cases} f_j(x-1) & \text{si } x > 0, \text{ et si } M_{i,j}, \text{ sur entrée } x \in \mathbb{N}, \text{ ne s'arrête pas en } x \text{ étapes ou moins,} \\ f_j(x-1) + y & \text{si } x > 0, \text{ et si } M_{i,j}, \text{ sur entrée } x \in \mathbb{N}, \text{ s'arrête en } x \text{ étapes ou moins, et retourne } y \in \mathbb{N}, \\ 0 & \text{sinon.} \end{cases}$$

Intuitivement, f_j est une fonction non décroissante qui accumule les sorties de $M_{i,j}$ sur entrée $x = 0, 1, \ldots$ Contrairement aux réseaux bien stucturés de dimension 1 de [FMP04], nous imposons x étapes de simulation afin que la classe soit effective.

Sans cette contrainte, il nous serait impossible de parler de décidabilité.

Puisque $\mathcal{F}_1$ est une classe de WSTS fonctionnels sur $\mathbb{N}$, par la proposition 2.17, $\mathcal{F}_1$ est une classe de WSTS-ω^2 avec monotonicité forte et bel ordre. De plus, chacune des fonctions f_j définies ci-dessus peut être calculée en exécutant $M_{i,j}$ pendant un nombre fini d'étapes. Ainsi, $\mathcal{F}_1$ est post-effective. Toutefois, $\mathcal{F}_1$ n'est ni post-effective sous complétion, ni pré-effective sous clôture par le haut :

Proposition 3.49. *$\mathcal{F}_1$ n'est pas effective sous complétion et n'est pas pré-effective sous clôture par le haut.*

Démonstration. Soit Turing'_i une machine de Turing qui, sur entrée $x \in \mathbb{N}$, exécute Turing_i sur son propre encodage et retourne 1. Soit $\mathcal{S}_i = (\mathbb{N}, \to, \{f_{\mathsf{Turing}'_i}\}, \leq)$ le WSTS de $\mathcal{F}_1$ avec l'unique fonction partielle $f_{\mathsf{Turing}'_i}$. Si Turing_i ne s'arrête pas sur son encodage, alors Turing'_i ne s'arrête sur aucun entrée, et ainsi $\mathsf{Succ}_{\mathcal{S}_i}(x) = \{0\}$ pour tout $x \in \mathbb{N}$. Si Turing_i s'arrête sur son encodage, alors il existe $m \in \mathbb{N}$ tel que $\mathsf{Succ}_{\mathcal{S}_i}(x) = \{0\}$ pour tout $x < m$ et $\mathsf{Succ}_{\mathcal{S}_i}(x) = \{x - m + 1\}$ pour tout $x \geq m$. Par définition de $\mathcal{S}_i$ et $\widehat{\mathcal{S}_i}$, nous avons

$$\begin{aligned}
\mathbb{N} \rightsquigarrow \mathbb{N} &\iff \mathbb{N} \subseteq {\downarrow}\mathsf{Succ}_{\mathcal{S}_i}(\mathbb{N}) \\
&\iff \mathsf{Succ}_{\mathcal{S}_i}(\mathbb{N}) = \mathbb{N} \\
&\iff \mathsf{Turing}'_i \text{ retourne 1 pour un nombre infini d'entrées } x \in \mathbb{N} \\
&\iff \mathsf{Turing}_i \text{ s'arrête sur son propre encodage.}
\end{aligned}$$

Si $\mathcal{F}_1$ était effective sous complétion, il serait possible de produire la décomposition $\mathsf{DecompIdeaux}(\overline{\emptyset}) = \{\mathbb{N}\}$ et de vérifier si $\mathbb{N} \rightsquigarrow \mathbb{N}$ dans $\mathcal{S}_i$. Cela donnerait lieu à une procédure décidant le problème d'arrêt. Ainsi, $\mathcal{F}_1$ n'est pas effective sous complétion.

Nous remarquons également que ${\uparrow}\mathsf{Pred}_{\mathcal{S}_i}({\uparrow}1) \neq \emptyset$ si, et seulement si, Turing_i s'arrête sur son encodage. Ainsi, si $\mathcal{F}_1$ était pré-effective sous clôtures par le haut, il serait possible de décider si ${\uparrow}\mathsf{Pred}_{\mathcal{S}_i}({\uparrow}1) \neq \emptyset$ et ainsi de décider si Turing_i s'arrête sur son encodage. Ainsi, $\mathcal{F}_1$ n'est pas pré-effective par le haut. □

Bien que $\mathcal{F}_1$ ne soit ni post-effective sous complétion, ni pré-effective sous clôture par le haut, son problème de couverture est tout de même décidable.

Proposition 3.50. *Le problème de couverture pour $\mathcal{F}_1$ est décidable.*

Démonstration. Soit $\mathcal{S} = (\mathbb{N}, \to, F, \leq) \in \mathcal{F}_1$ où $F = \{f_1, f_2, \ldots, f_n\}$. Soient $x, y \in \mathbb{N}$. Notons d'abord que si $x \geq y$, alors y est trivialement couvrable à partir de x et nous avons terminé. Supposons donc que $x < y$.

Posons $F^{\leq \ell} \stackrel{\text{déf}}{=} \{f_{i_1} \circ f_{i_2} \circ \cdots \circ f_{i_k} : k \leq \ell, i_1, i_2, \ldots, i_k \in [n]\}$ et $m_\ell \stackrel{\text{déf}}{=} \max\{g(x) : g \in F^{\leq \ell}\}$. Considérons la séquence infinie $m_0, m_1, \ldots$, et montrons que si deux termes consécutifs sont égaux, alors la séquence se stabilise. Supposons que $m_{\ell+1} = m_\ell$ pour un certain $\ell \in \mathbb{N}$, alors $f_i(m_\ell) \leq m_\ell$ pour tout $i \in [n]$. Ainsi, par monotonicité et par induction sur p, nous pouvons prouver que $g(m_\ell) \leq m_\ell$ pour tout $g \in F^{\leq p}$. Ainsi, $m_{\ell+p} \leq m_\ell$ pour tout $p \in \mathbb{N}$. Par construction, $m_{\ell+p} \geq \ell_n$. Par conséquent, $m_{\ell+p} = m_\ell$ et la séquence se stabilise donc.

Puisque la séquence croît strictement jusqu'à ce qu'elle se stabilise, il est suffisant de calculer m_{y-x}. Si $m_{y-x} \geq y$, alors y est couvrable à partir de x, sinon ce n'est pas le cas. □

3.4 Discussion

Dans ce chapitre, nous avons dressé un portait de la décidabilité du problème de terminaison, du problème de finitude, du problème de maintenabilité et du problème de couverture pour les WSTS. Nous avons étendu les résultats de décidabilité aux WSTS à branchement infini en tirant profit de la décomposition finie d'ensembles clos par le bas en idéaux et de la complétion de WSTS. Nous avons également établi une distinction entre les variantes faibles et fortes des problèmes de terminaison et de maintenabilité. Ces résultats sont résumés au tableau récapitulatif 3.I.

Nous avons simplifié la notion de complétion de WSTS introduite dans [FG12], de telle sorte qu'elle ne dépend plus de notion algébrique et/ou topologique, et qu'elle s'applique aux WSTS à branchement infini. Notre étude de la complétion

	Branchement fini	Branchement infini
Terminaison	Déc. • post-effectivité • monotonicité transitive	Indéc. • post-effectivité • post-effectivité sous complétion • monotonicité forte et stricte • WSTS-ω^2 • bel ordre
	Indéc. • post-effectivité • WSTS-ω^2 • bel ordre	
Terminaison forte	Déc. • post-effectivité • monotonicité transitive	Déc. • post-effectivité sous complétion • monotonicité transitive • WSTS-ω^2
	Indéc. • post-effectivité • WSTS-ω^2 • bel ordre	
Finitude	Déc. • post-effectivité • monotonicité stricte • bel ordre	
	Indéc. • post-effectivité • post-effectivité sous complétion • WSTS-ω^2 • monotonicité forte • bel ordre	
Maintenabilité	Déc. • post-effectivité • monotonicité bégayante	Indéc. • post-effectivité • post-effectivité sous complétion • monotonicité forte et stricte • WSTS-ω^2 • bel ordre
	Indéc. • post-effectivité • WSTS-ω^2 • bel ordre	
Maintenabilité faible	Déc. • post-effectivité • monotonicité bégayante	Déc. • post-effectivité sous complétion • monotonicité forte • WSTS-ω^2
	Indéc. • post-effectivité • WSTS-ω^2 • bel ordre	
Couverture	Déc. • pré-effectivité sous clôture par le haut	
	Déc. • post-effectivité sous complétion	
	Indéc. • post-effectivité • monotonicité forte et stricte • WSTS-ω^2 • bel ordre	

Tableau 3.I – Décidabilité (en vert) et indécidabilité (en rouge) des problèmes de décisions pour les classes de WSTS. Les régions hachurées et foncées identifient les nouveaux résultats établis dans cette thèse.

de WSTS a établi par des correspondances entre les exécutions, la mononoticité et l'effectivité d'un WSTS et celles de sa complétion.

La complétion de WSTS-ω^2 ouvre la voie à la conception de nouveaux algorithmes avant. Ces algorithmes sont souvent conceptuellement plus simples que les algorithmes arrière et, à certains égards, peuvent s'avérer plus efficaces. Il sera intéressant d'étudier plus en profondeur les aspects algorithmiques des outils dé-

veloppés dans ce chapitre. En particulier, comment se comparent l'efficacité des méthodes avant et arrière sur des modèles concrets de WSTS ?

4

SUR LA COMPLEXITÉ DU PROBLÈME D'ACCESSIBILITÉ POUR LES SYSTÈMES D'ADDITION DE VECTEURS

Dans ce chapitre, nous nous penchons sur la complexité du problème d'accessibilité pour les systèmes d'addition de vecteurs :

ACCESSIBILITÉ

Entrée: Un d-VASS $\mathcal{V} = (Q, T)$ et $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^d$.
Question: $p(\boldsymbol{u}) \to^*_{\mathbb{N}^d} q(\boldsymbol{v})$?

La décidabilité et la complexité de ce problème sont étudiées depuis plus de quarante ans entre autres parce que plusieurs problèmes notables s'y réduisent. Avant que sa décidabilité ou son indécidabilité n'ait été établie, Lipton montra en 1976 que le problème est EXPSPACE-difficile [Lip76]. En 1977, Sacerdote & Tenney annonçaient au colloque STOC 1977 une preuve que le problème était décidable [ST77]. Leur preuve était toutefois incomplète, et tel que rapporté par Lipton [Lip09], certains des lemmes y étaient erronés même en dimension $d = 1$. En 1979, Hopcroft & Pansiot démontrèrent que le problème est décidable pour les 2-VASS [HP79] sans établir une quelconque borne de complexité. Le problème fut éventuellement montré décidable, et ce, indépendemment du nombre de compteurs. La première preuve valide de ce résultat est attribuée à Mayr [May81]. La complexité de cette preuve amena vraisemblablement Kosaraju [Kos82] à donner, l'année suivante, une nouvelle preuve de la décidabilité du problème, toujours sans borne de complexité. Quelques années plus tard, Howell *et al.* [HRHY86] analysèrent l'algorithme de Hopcroft & Pansiot, ce qui leur permit d'établir que le problème est soluble en temps doublement exponentiel pour les 2-VASS. Les preuves de

décidabilité de Mayr et Kosaraju furent revisitées par Lambert [Lam92], et de nouvelles preuves plus simples furent établies par Leroux dans une série d'articles successifs [Ler09, Ler11, Ler12]. Une première borne de complexité pour le problème générale fut établie en 2015 par Leroux & Schmitz [LS15]. Ils montrèrent que le problème se situe dans la classe F_{ω^3}, une classe qui comprend des fonctions non primitives récursives telles que la fonction d'Ackermann. Cette borne fut obtenue en remarquant que l'algorithme, dit KLMST[1], travaille dans une certaine complétion au sens du chapitre 3. Dans la même année, la complexité précise du problème fut établie pour les 2-VASS par l'auteur et Finkel, Göller, Haase & McKenzie [BFG+15]. Nous dévouons ce chapitre à ce dernier résultat, c'est-à-dire, à la PSPACE-complétude du problème d'accessibilité pour les 2-VASS. Nous discuterons également brièvement de ses implications. Notons que le lecteur peut se référer à [Sch16] pour un survol des résultats de complexité de [LS15] et [BFG+15].

Le problème d'accessibilité pour les 2-VASS est facilement vu comme étant PSPACE-difficile grâce aux travaux récents de Fearnley & Jurdziński [FJ15]. Ainsi, dans ce chapitre nous montrons que le problème est dans PSPACE. Pour parvenir à ce résultat, nous bornons la taille des exécutions de longueur minimale des 2-VASS, et ainsi des exécutions témoignant de l'accessibilité. Nous montrons que la taille de ces exécutions est exponentielle en réexplorant les travaux de Leroux & Sutre [LS04] qui montrent que la relation d'accessibilité d'un 2-VASS est décrite par des langages réguliers bornés de la forme $\alpha_0\beta_1^*\alpha_1\cdots\beta_k^*\alpha_k$. Nous bornons la taille de ces langages et montrons comment cela mène à des bornes sur la taille des exécutions.

À la section 4.1, nous introduisons les notions et définitions propres à ce chapitre. Ensuite, à la section 4.2, nous donnons un aperçu plus précis de la stratégie de preuve qui sera suivie. Aux sections 4.3 et 4.4, nous montrons comment la relation d'accessibilité des 1-VASS et des 2-VASS peut être décrite par des langages réguliers bornés. Nous terminons le chapitre, à la section 4.5, en montrant que le problème d'accessibilité pour les 2-VASS est PSPACE-complet. Nous discutons

[1]Pour Kosaraju, Lambert, Mayr, Sacerdote & Tenney.

d'autres conséquences de nos résultats. En particulier, nous montrons que lorsque les entiers sont encodés en notation unaire, le problème est NL-difficile et dans NP.

4.1 Préliminaires

Soit $\mathcal{V} = (Q, T)$ un d-VASS. Soit $\pi = (p_1, \boldsymbol{z}_1, q_1)(p_2, \boldsymbol{z}_2, q_2) \cdots (p_k, \boldsymbol{z}_k, q_k) \in T^k$. Nous disons que π est un *chemin, de* p_1 *vers* q_k, si $p_i = q_{i+1}$ pour tout $i \in [k-1]$. Nous dénotons l'ensemble des chemins de $\mathcal{V}$ par $\mathsf{Chemins}_{\mathcal{V}}$, et l'ensemble des chemins de p vers q par $\mathsf{Chemins}_{\mathcal{V}}(p, q)$. Un chemin $\pi \in \mathsf{Chemins}_{\mathcal{V}}(p, q)$ est un *cycle* si $|\pi| \geq 1$ et $p = q$. Un chemin $\pi \in \mathsf{Chemins}_{\mathcal{V}}$ est *acyclique* si aucun infixe de π est un cycle. Un cycle π est *simple* si π est l'unique infixe de π qui soit un cycle. Un *schéma linéaire (de chemins)* ρ, de $p \in Q$ vers $q \in Q$, est une expression régulière[2]

$$\rho \;=\; \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k,$$

telle que $\alpha_0 \beta_1 \alpha_1 \cdots \beta_k \alpha_k$ est un chemin, de p vers q, et telle que β_i est un cycle, pas nécessairement simple, pour tout $i \in [k]$. Nous définissons la *longueur* de ρ par $|\rho| \stackrel{\text{déf}}{=} |\alpha_0 \beta_1 \alpha_1 \cdots \beta_k \alpha_k|$ et son *nombre de cycles* par $|\rho|_* \stackrel{\text{déf}}{=} k$. Nous appelons $\beta_1, \beta_2 \ldots, \beta_k$ *les cycles de* ρ. Notons que tout chemin π induit un schéma linéaire en posant $\alpha_0 = \pi$ et $k = 0$. Nous illustrons la structure générale d'un schéma linéaire à la figure 4.1.

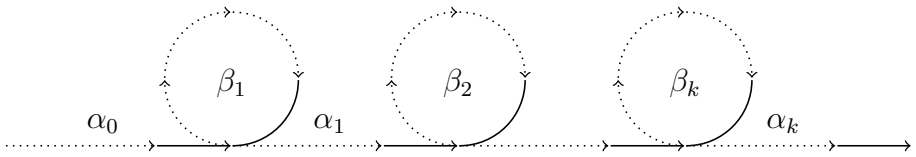


Figure 4.1 – Illustration d'un schéma linéaire $\rho = \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$.

Soit S un ensemble de schémas linéaires, nous écrivons $S(p, q)$ afin de dénoter le sous-ensemble de schémas linéaires de p vers q, c.-à-d., $S(p, q) \stackrel{\text{déf}}{=} \{\rho \in S : \rho \text{ est un schéma linéaire de } p \text{ vers } q\}$. Soient $\mathbb{A} \subseteq \mathbb{B} \subseteq \mathbb{Z}^d$ et S un ensemble fini de

[2]Nous faisons référence au langage associé à un schéma linéaire ρ de façon implicite. Ainsi, ρ désigne à la fois la description d'un langage régulier, et le langage régulier lui-même.

schémas linéaires, nous disons que S *capture* $\rightarrow^*_{\mathbb{A}}$ *dans* $\mathbb{B}$ si pour tous $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{A}$, $p(\boldsymbol{u}) \rightarrow^*_{\mathbb{A}} q(\boldsymbol{v}) \implies p(\boldsymbol{u}) \xrightarrow{S}_{\mathbb{B}} q(\boldsymbol{v})$. La relation d'accessibilité $\rightarrow^*_{\mathbb{A}}$ est dite *plate pour* $\mathbb{B}$ s'il existe un ensemble fini S de schémas linéaires qui la capture dans $\mathbb{B}$. Par extension, un d-VASS $\mathcal{V}$ est dit *aplatissable dans* $\mathbb{B}$ si sa relation $\rightarrow^*_{\mathbb{N}^d}$ est plate pour $\mathbb{B}$. Dans chacun des cas, lorsque nous ne spécifions pas $\mathbb{B}$, il est sous-entendu que $\mathbb{B} = \mathbb{N}^d$.

Soit $\mathcal{V} = (Q, T)$ un d-VASS. Le *déplacement* d'une transition $t = (p, \boldsymbol{z}, q) \in T$ est défini par $\delta(t) \stackrel{\text{déf}}{=} \boldsymbol{z}$. De façon plus générale, le *déplacement* d'une suite de transitions $t_1, t_2, \ldots, t_k \in T$ est défini par $\delta(t_1 t_2 \cdots t_k) \stackrel{\text{déf}}{=} \delta(t_1) + \delta(t_2) + \ldots + \delta(t_k)$, et $\delta(\varepsilon) = 0$. Pour chaque schéma linéaire $\rho = \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$, nous dénotons $\mathsf{Cycles}(\rho) \stackrel{\text{déf}}{=} \{\delta(\beta_1), \delta(\beta_2), \ldots, \delta(\beta_k)\}$ l'ensemble des déplacements des cycles de ρ.

Nous écrivons $\|T\|$ afin de représenter la plus grande valeur apparaissant sur l'une des transitions de $\mathcal{V}$, plus précisément $\|T\| \stackrel{\text{déf}}{=} \max\{\|\delta(t)\| : t \in T\}$. Nous considérons que $\mathcal{V}$ est naturellement encodé comme un graphe dirigé et étiqueté, p. ex. par une liste ou une matrice d'adjacence où les entrées sont les déplacements des transitions. Notons que lorsque les entiers contenus dans les vecteurs apparaissant sur les transitions sont encodés en binaire, $\|T\|$ est exponentiel dans la taille de l'encodage de $\mathcal{V}$, alors que $\|T\|$ est polynomial lorsque les entiers sont encodés en notation unaire.

Un *quadrant* est un ensemble parmi $\mathbb{N} \times \mathbb{N}$, $-\mathbb{N} \times \mathbb{N}$, $-\mathbb{N} \times -\mathbb{N}$ et $\mathbb{N} \times -\mathbb{N}$.

4.2 Stratégie globale de preuve

Nous donnons un aperçu de la stratégie de preuve utilisée dans ce chapitre. Afin d'établir la complexité du problème d'accessibilité pour les 2-VASS, nous montrons que tous les 2-VASS sont aplatissables, c.-à-d., que la relation d'accessiblité $\rightarrow^*_{\mathbb{N}^2}$ est capturée par un ensemble fini de schémas linéaires. Par exemple, considérons le 2-VASS $\mathcal{V}$ illustré à la figure 4.2. Celui-ci contient des cycles qui ne sont pas simples. Par exemple, le cycle $t_1\, t_2\, t_3$ contient le cycle t_2. Néanmoins, en analysant attentivement $\mathcal{V}$, nous pouvons montrer que chaque chemin est « équivalent » à un

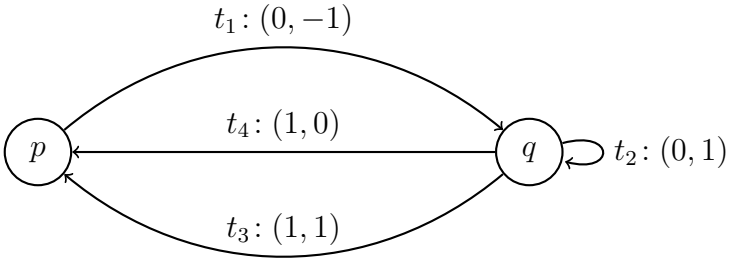


Figure 4.2 – Exemple de 2-VASS.

chemin provenant d'un schéma linéaire, au sens suivant :

$$p(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v}) \iff p(\boldsymbol{u}) \xrightarrow{t_1 t_2^* (t_3 t_1)^* (t_4 t_1)^*} \mathbb{N}^2 \; q(\boldsymbol{v})$$

$$p(\boldsymbol{u}) \to^*_{\mathbb{N}^2} p(\boldsymbol{v}) \iff p(\boldsymbol{u}) \xrightarrow{t_1 t_2^* (t_3 t_1)^* (t_4 t_1)^* t_3 \cup t_1 t_2^* (t_3 t_1)^* (t_4 t_1)^* t_4 \cup \varepsilon} \mathbb{N}^2 \; p(\boldsymbol{v})$$

$$q(\boldsymbol{u}) \to^*_{\mathbb{N}^2} p(\boldsymbol{v}) \iff q(\boldsymbol{u}) \xrightarrow{t_2^* (t_3 t_1)^* (t_4 t_1)^* t_3 \cup t_2^* (t_3 t_1)^* (t_4 t_1)^* t_4} \mathbb{N}^2 \; p(\boldsymbol{v})$$

$$q(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v}) \iff q(\boldsymbol{u}) \xrightarrow{t_2^* (t_3 t_1)^* (t_4 t_1)^*} \mathbb{N}^2 \; q(\boldsymbol{v})$$

La relation $\to^*_{\mathbb{N}^2}$ de $\mathcal{V}$ est donc capturée par un ensemble de schémas linéaires. Ainsi, par exemple, pour tester si $p(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v})$, il suffit de vérifier s'il existe $e_1, e_2, e_3 \in \mathbb{N}$ tels que $p(\boldsymbol{u}) \xrightarrow{t_1 t_2^{e_1} (t_3 t_1)^{e_2} (t_4 t_1)^{e_3}} \mathbb{N}^2 \; q(\boldsymbol{v})$. Bien que cette vérification ne soit pas complètement triviale, elle s'avère beaucoup plus simple qu'un test d'accessibilité dans sa forme générale.

Nous montrons que tous les 2-VASS sont aplatissables et, plus précisément, que leur relation d'accessibilité $\to^*_{\mathbb{N}^2}$ est capturée par des « petits » schémas linéaires. Plus formellement, nous montrons que pour tout 2-VASS $\mathcal{V} = (Q, T)$, la relation d'accessibilité $\to^*_{\mathbb{N}^2}$ est capturée par un ensemble fini de schémas linéaires chacun de taille polynomiale en $|Q| + \|T\|$ et possédant un nombre quadratique de cycles en fonction de $|Q|$:

Théorème 4.1. *Soit $\mathcal{V} = (Q, T)$ un 2-VASS. Il existe un ensemble fini S de*

schémas linéaires tel que[3]

- *pour tous* $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^2$, $p(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v}) \iff p(\boldsymbol{u}) \xrightarrow{S}_{\mathbb{N}^2} q(\boldsymbol{v})$, *et*
- $|\rho| \leq (|Q| + \|T\|)^{O(1)}$ *et* $|\rho|_* \leq O(|Q|^2)$ *pour tout* $\rho \in S$.

Lorsque nous aurons établi le théorème 4.1, nous pourrons réduire le problème d'accessibilité à la vérification de l'existence d'une solution d'un système d'inégalités diophantiennes construit à partir de S. En appliquant des bornes standards tirées de la programmation linéaire entière, cela nous permettra d'établir une borne exponentielle sur la longueur de chemins témoignant de l'accessibilité, et ainsi d'obtenir le résultat principal de ce chapitre :

Théorème 4.2. *Le problème d'accessibilité pour les 2-VASS est* PSPACE*-complet.*

Afin de prouver le théorème 4.1, nous montrons d'abord que les 1-VASS sont aplatissables grâce à des schémas linéaires de taille appropriée. Cela nous permettra par la suite de montrer que les 2-VASS sont aplatissables en analysant plusieurs types d'exécutions. Nous pourrons ensuite prouver le théorème 4.2.

4.3 Aplatissement des 1-VASS

Dans cette section, nous montrons que tous les 1-VASS sont aplatissables, dans le but de prouver plus généralement, à la section suivante, que tous les 2-VASS sont aplatissables. Bien que les 1-VASS sont déjà connus comme étant aplatissables, il n'existe pas de borne explicite sur la taille des schémas linéaires dans la littérature. Ainsi, nous construisons des schémas linéaires dont nous bornons la taille et le nombre de cycles.

Formellement, nous démontrerons la proposition suivante.

[3]La signification plus technique de cet énoncé est qu'il existe des constantes c_1 et c_2 telles que pour tout 2-VASS $\mathcal{V} = (Q, T)$, il existe un ensemble fini S de schémas linéaires tel que pour tout $\rho \in S$, $|\rho| \leq (|Q| + \|T\|)^{c_1}$ et $|\rho|_* \leq c_2|Q|^2$, avec la propriété additionnelle que pour tout $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^2$, $p(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v}) \iff p(\boldsymbol{u}) \xrightarrow{S}_{\mathbb{N}^2} q(\boldsymbol{v})$. Nous éviterons ce niveau de formalisme afin de faciliter la lecture et d'éviter de préciser les valeurs exactes des constantes qui seront très grandes.

Proposition 4.3. *Soit $\mathcal{V} = (Q, T)$ un 1-VASS. Il existe un ensemble fini Y de schémas linéaires tel que pour tous $p(u), q(v) \in Q \times \mathbb{N}$,*

(i) $p(u) \to^*_{\mathbb{N}} q(v) \iff p(u) \xrightarrow{Y}_{\mathbb{N}} q(v)$,

(ii) $|\rho| \leq (|Q| + \|T\|)^{O(1)}$ *et* $|\rho|_* \leq 1$ *pour tout* $\rho \in Y$.

Afin de prouver la proposition précédente, nous ferons appel au lemme suivant établi par Valiant et Paterson [VP75], qui montre indirectement que la relation $\to^*_{\mathbb{N}}$ d'un 1-VASS est capturée par un ensemble fini de schémas linéaires possédant chacun au plus un cycle.

Lemme 4.4 ([VP75, lemme 2])**.** *Soit $\mathcal{V} = (Q, T)$ un 1-VASS tel que $T \subseteq Q \times \{-1, 0, 1\} \times Q$. Soient $p(u), q(v) \in Q \times \mathbb{N}$ tels que $p(u) \to^*_{\mathbb{N}} q(v)$ et $|v-u| \geq |Q|+|Q|^2$. Il existe $\pi \in \mathsf{Chemins}_{\mathcal{V}}(p, q)$ et $\alpha, \beta, \gamma \in T^*$ tels que $p(u) \xrightarrow{\pi}_{\mathbb{N}} p(v)$, et*

(i) $\pi = \alpha\beta^i\gamma$ *pour un certain* $i \in \mathbb{N}_{>0}$,

(ii) $\alpha\beta^*\gamma$ *est un schéma linéaire,*

(iii) $|\alpha\gamma| < |Q|^2$ *et* $1 \leq |\beta|, |\delta(\beta)| \leq |Q|$.

Le lemme précédent nous permet de borner la taille des exécutions dans les 1-VASS dont les transitions modifient le compteur de -1, 0 ou 1 comme suit :

Lemme 4.5. *Soit $\mathcal{V} = (Q, T)$ un 1-VASS tel que $T \subseteq Q \times \{-1, 0, 1\} \times Q$. Soient $p(u), q(v) \in Q \times \mathbb{N}$ et $D \stackrel{déf}{=} |v - u|$. Si $p(u) \to^*_{\mathbb{N}} q(v)$, alors $p(u) \xrightarrow{\pi}_{\mathbb{N}} q(v)$ pour un certain $\pi \in \mathsf{Chemins}_{\mathcal{V}}(p, q)$ tel que $|\pi| \leq |Q|^{O(1)} + |Q| \cdot D$.*

Démonstration. Nous considérons d'abord le cas où $D \geq |Q|+|Q|^2$. Par le lemme 4.4, nous avons $p(u) \xrightarrow{\alpha\beta^i\gamma}_{\mathbb{N}} q(v)$ pour un certain $i \geq 0$, où $\alpha\beta^*\gamma$ est un schéma linéaire, $|\alpha\gamma| < |Q|^2$ et β est un cycle tel que $1 \leq |\beta|, |\delta(\beta)| \leq |Q|$. Puisque $1 \leq |\delta(\beta)| \leq |Q|$, nous avons $i \leq D + |\alpha\gamma|$ et ainsi $|\alpha\beta^i\gamma| < |Q|^2 + |Q| \cdot i \leq |Q|^{O(1)} + |Q| \cdot D$.

Nous considérons maintenant le cas où $D < |Q| + |Q|^2$. Si $p(u) \to^*_{\mathbb{N}} q(v)$, alors $p(u) \xrightarrow{\pi} q(v)$ où $|\pi|$ est minimal et l'une des deux propriétés suivante est satisfaite :

(i) chaque configuration intermédiaire $r(w)$ est telle que $|w-v| < 2\cdot(|Q|+|Q|^2)$,

(ii) il existe une configuration intermédiaire $r(w)$ telle que $|w-v| = 2\cdot(|Q|+|Q|^2)$.

Toute exécution vérifiant (i) est de longueur inférieure à $4 \cdot (|Q| + |Q|^2) \cdot |Q|$ par minimalité de $|\pi|$. Toute exécution vérifiant (ii) comprend une configuration intermédiaire $r(w)$ telle que $|w - v| = 2 \cdot (|Q| + |Q|^2)$. Par l'inégalité triangulaire, $|w-v| \leq |w-u|+|u-v|$. Nous concluons que $|Q|+|Q|^2 \leq |w-u| \leq 3\cdot(|Q|+|Q|^2)$, car

$$\begin{aligned} |Q| + |Q|^2 &\leq |w - v| - |u - v| \\ &\leq |w - u| \\ &\leq |w - v| + |u - v| \\ &\leq 3 \cdot (|Q| + |Q|^2) \end{aligned}$$

où la première et la dernière inégalité utilisent $|v-u| = D < |Q|+|Q|^2$. Ainsi, nous avons $|Q|+|Q|^2 \leq |w-u| \leq 3\cdot(|Q|+|Q|^2)$ et $|w-v| = 2\cdot(|Q|+|Q|^2)$. Par conséquent, $|\pi|$ est borné par la longueur de deux exécutions minimales respectivement de $p(u)$ à $r(w)$, et de $r(w)$ à $q(v)$, où $|w - u| \geq |Q| + |Q|^2$ et $|v - w| \geq |Q| + |Q|^2$. Cela nous ramène donc au cas traité en début de preuve, et ainsi,

$$\begin{aligned} |\pi| &\leq (|Q|^2 + |Q| \cdot 3 \cdot (|Q| + |Q|^2)) + (|Q|^2 + |Q| \cdot 2 \cdot (|Q| + |Q|^2)) \\ &\leq |Q|^{O(1)} \ . \end{aligned}$$

□

Nous sommes maintenant en mesure de combiner les lemmes 4.4 et 4.5 afin de montrer la proposition principale de cette section, soit, que la relation $\rightarrow^*_{\mathbb{N}}$ d'un 1-VASS est capturée par un ensemble fini de schémas linéaires chacun de taille polynomiale en $|Q| + \|T\|$ et possédant au plus un cycle.

Preuve de la proposition 4.3. Nous construisons un 1-VASS $\mathcal{V}' = (Q', T')$, tel que $T' \subseteq Q' \times \{-1, 0, +1\} \times Q'$, qui possède les mêmes comportements que $\mathcal{V}$. Nous

pourrons ainsi appliquer les lemmes 4.4 et 4.5 à $\mathcal{V}'$.

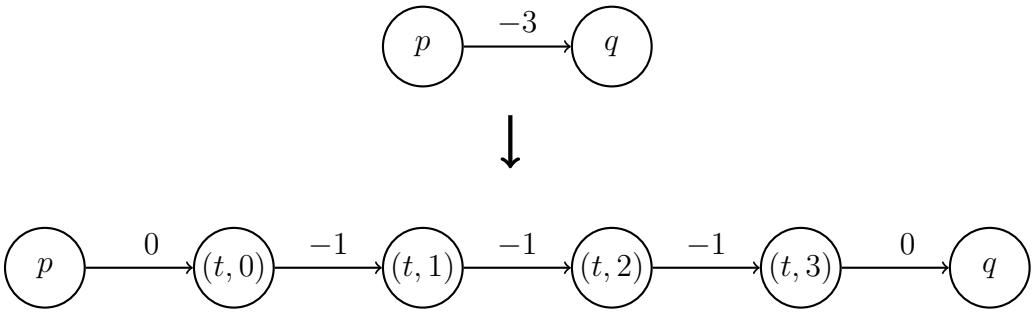


Figure 4.3 – Exemple de transformation d'une transition (p, z, q) en une suite équivalente de transitions de déplacement $\{-1, 0, 1\}$.

Nous associons à chaque transition $t = (q, z, q') \in T$ une séquence de $|z| + 2$ transitions dans $\mathcal{V}'$, dont $|z|$ d'entres elles incrémentent ou décrémentent le compteur. Cette transformation est illustrée à la figure 4.3. Posons $\mathsf{val}(z) = 1$ si $v \geq 0$ et $\mathsf{val}(z) = -1$ si $v < 0$. Plus formellement, nous définissons $\mathcal{V}'$ de la façon suivante,

$$Q' \stackrel{\text{déf}}{=} Q \cup \{(t, i) : t = (p, z, q) \in T, i \in [0, |z|]\}$$

et

$$\begin{aligned} T' \stackrel{\text{déf}}{=} & \quad \{(p, 0, (t, 0)) : t = (p, z, q) \in T\} \\ & \cup \{((t, |z|), 0, q) : t = (p, z, q) \in T\} \\ & \cup \{((t, i), \mathsf{val}(z), (t, i + 1)) : t = (p, z, q) \in T, 0 \leq i < z\} \ . \end{aligned}$$

Nous définissons un morphisme $h : T \to T'^*$ entre les transitions de $\mathcal{V}$ et $\mathcal{V}'$. Pour tout $t = (q, z, q') \in T$, h est défini par

$$h(t) \stackrel{\text{déf}}{=} (q, 0, (t, 0)) \cdot \left(\prod_{i=1}^{|z|} ((t, i - 1), \mathsf{val}(z), (t, i)) \right) \cdot ((t, |z|), 0, q') \ .$$

On voit que toute exécution dans $\mathcal{V}$ correspond, sous h, à une exécution dans $\mathcal{V}'$ et vice-versa. Plus formellement, il est possible de montrer que :

(i) $|h(t)| \leq \|T\| + 2$ pour tout $t \in T$,

(ii) si $p(u) \xrightarrow{\pi}_{\mathbb{N}} q(v)$ dans $\mathcal{V}$, alors $p(u) \xrightarrow{h(\pi)}_{\mathbb{N}} q(v)$ dans $\mathcal{V}'$, et

(iii) si $p, q \in Q$ et $p(u) \xrightarrow{\pi'}_{\mathbb{N}} q(v)$ dans $\mathcal{V}'$, alors il existe un unique chemin $\pi \in \mathsf{Chemins}_{\mathcal{V}}(p, q)$ tel que $\pi' = h(\pi)$ et $p(u) \xrightarrow{\pi}_{\mathbb{N}} q(v)$ dans $\mathcal{V}$.

Soient $p(u), q(v) \in Q \times \mathbb{N}$, et $\pi' \in \mathsf{Chemins}_{\mathcal{V}'}(p, q)$ tel que $p(u) \xrightarrow{\pi'}_{\mathbb{N}} q(v)$ dans $\mathcal{V}'$. Par le point (iii), nous pouvons écrire $h^{-1}(\pi')$ afin de dénoter l'unique $\pi \in \mathsf{Chemins}_{\mathcal{V}}(p, q)$ tel que $h(\pi) = \pi'$ et $p(u) \xrightarrow{\pi}_{\mathbb{N}} q(v)$ dans $\mathcal{V}$. En particulier, π' est un cycle dans $\mathcal{V}'$ si, et seulement si, $h^{-1}(\pi')$ est un cycle dans $\mathcal{V}$.

Afin de montrer l'existence d'un ensemble fini de schémas linéaires tel que décrit dans l'énoncé du lemme, nous montrons que lorsque $p(u) \rightarrow^*_{\mathbb{N}} q(v)$ dans $\mathcal{V}$, il existe un schéma linéaire $\rho \subseteq T^*$ tel que $p(u) \xrightarrow{\rho}_{\mathbb{N}} q(v)$ dans $\mathcal{V}$, et tel que $|\rho| \leq (|Q| + \|T\|)^{O(1)}$. Soit $D = |Q'| + |Q'|^2$. Supposons que $p(u) \rightarrow^*_{\mathbb{N}} q(v)$ dans $\mathcal{V}$. Par le point (ii), nous avons $p(u) \rightarrow^*_{\mathbb{N}} q(v)$ dans $\mathcal{V}'$. Nous terminons la preuve en distinguant parmi deux cas : $|u - v| \leq D$ et $|u - v| > D$.

Cas 1 : $|u - v| \leq D$. Par le lemme 4.5, nous avons $p(u) \xrightarrow{\pi'}_{\mathbb{N}} q(v)$ dans $\mathcal{V}'$ pour un certain $\pi' \in \mathsf{Chemins}_{\mathcal{V}'}(p, q)$ tel que $|\pi'| \leq |Q'|^{O(1)} + |Q'| \cdot D \leq (|Q| + \|T\| + D)^{O(1)}$. Ainsi, nous posons $\rho = h^{-1}(\pi')$ et notons que $p(u) \xrightarrow{\rho}_{\mathbb{N}} q(v)$ dans $\mathcal{V}$ par le point (iii). De plus $|\rho| \leq |\pi'| \leq (|Q| + \|T\| + D)^{O(1)} \leq (|Q| + \|T\|)^{O(1)}$ tel que requis.

Cas 2 : $|u - v| > D$. Par le lemme 4.4, nous avons $p(u) \xrightarrow{\alpha'(\beta')^i\gamma'}_{\mathbb{N}} q(v)$ dans $\mathcal{V}'$ pour $i \in \mathbb{N}_{>0}$ et un certain schéma linéaire $\rho' = \alpha'\beta'^*\gamma'$ satisfaisant $|\alpha'\gamma'| < |Q'|^2$ et $|\beta'| \leq |Q'|$. Soit $q' \in Q'$ le premier état du cycle β'.

Si $q' \in Q$, alors nous avons $\alpha' \in \mathsf{Chemins}_{\mathcal{V}'}(p, q')$, $\beta' \in \mathsf{Chemins}_{\mathcal{V}'}(q', q')$ et $\gamma' \in \mathsf{Chemins}_{\mathcal{V}'}(q', q)$. Ainsi, par le point (iii),

$$\rho \stackrel{\text{déf}}{=} \alpha'^{-1}(\beta'^{-1})^*\gamma'^{-1}$$

est un schéma linéaire tel que $p(u) \xrightarrow{\rho}_{\mathbb{N}} q(v)$ dans $\mathcal{V}$. De plus, $|\rho| \leq |Q'| + |Q'|^2 \leq (|Q| + \|T\|)^{O(1)}$.

Autrement, si $q' \in Q' \setminus Q$, nous avons $q' = (t, i)$ pour un certain $t = (q_1, z, q_2) \in T$ et un certain $1 \leq i \leq |z|$. Puisque β' est un cycle de (t, i) vers (t, i), par définition de $\mathcal{V}'$, nous avons $\alpha' = \alpha'_1\alpha'_2$ et $\beta' = \beta'_1\beta'_2$, où $\alpha'_2 = \beta'_2 = (q_1, 0, (t, 0)) \cdot \prod_{j=1}^{i}((t, j-1), \mathsf{val}(z), (t, j))$. Nous avons donc

$$\alpha'(\beta')^*\gamma' \;=\; \alpha'_1\alpha'_2(\beta'_1\beta'_2)^*\gamma' \;=\; \alpha'_1(\beta'_2\beta'_1)^*\beta'_2\gamma' \;.$$

De plus, $\alpha'_1 \in \mathsf{Chemins}_{\mathcal{V}'}(p, q_1)$, $\beta'_2\beta'_1 \in \mathsf{Chemins}_{\mathcal{V}'}(q_1, q_1)$ et $\beta'_2\gamma' \in \mathsf{Chemins}_{\mathcal{V}'}(q_1, q)$. Ainsi, par le point (iii),

$$\rho \stackrel{\text{déf}}{=} h(\alpha'_1)^{-1}(h(\beta'_2\beta'_1)^{-1})^*h(\beta'_2\gamma')^{-1}$$

est un schéma linéaire tel que $p(u) \xrightarrow{\rho}_{\mathbb{N}} q(v)$ dans $\mathcal{V}$. De plus, $|\rho| \leq |Q'| + |Q'|^2 \leq (|Q| + \|T\|)^{O(1)}$. □

4.4 Aplatissement des 2-VASS

Dans cette section, nous prouvons le théorème 4.1. Afin de construire un ensemble fini de schémas linéaires pour chaque 2-VASS, nous analyserons certains types d'exécutions. Intuitivement, nous étudierons

(1) les exécutions qui demeurent près des axes, c.-à-d., dans $\mathbb{L} = ([0, D] \times \mathbb{N}) \cup (\mathbb{N} \times [0, D])$ pour un certain $D \in \mathbb{N}$.

(2) les exécutions qui demeurent loin des axes, c.-à-d., dans $\mathbb{O} = [D', \infty)^2$ pour un certain $D' \in \mathbb{N}$, et

Nous montrerons comment construire des schémas linéaires pour $\to^*_{\mathbb{O}}$ et $\to^*_{\mathbb{L}}$. Afin de combiner finiment ces schémas linéaires, dans le but de capturer $\to^*_{\mathbb{N}^2}$, nous devrons également étudier un troisième type d'exécutions :

(3) les exécutions qui débutent et terminent dans un même état de contrôle, et loin des axes, c.-à-d., dans $\mathbb{O}$, mais qui peuvent se rapprocher des axes, c.-à-d., entrer dans $\mathbb{L}$.

Nous illustrons ces trois types d'exécutions à la figure 4.4.

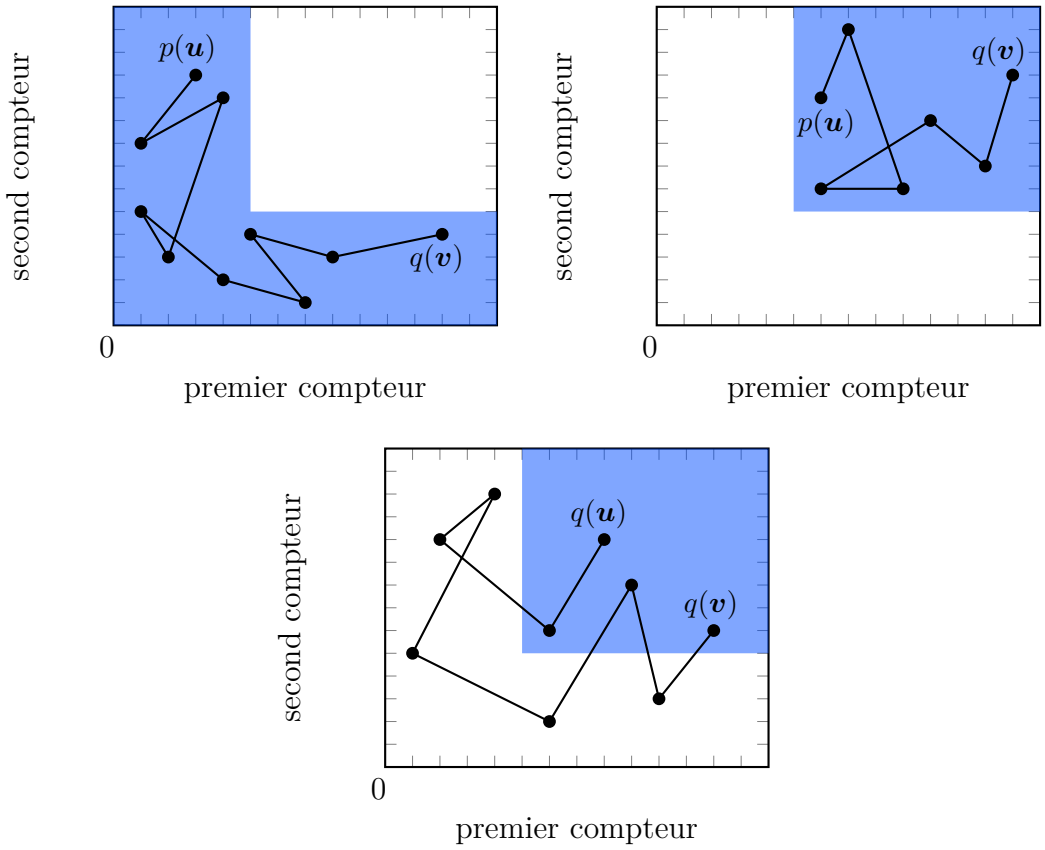


Figure 4.4 – Illustration des trois types d'exécutions étudiées. La région illustrée dans chaque cas est le quadrant positif du plan cartésien. *(1) Coin supérieur gauche :* exécution demeurant près des axes ; *(2) coin supérieur droit :* exécution demeurant loin des axes ; *(3) bas :* exécution de q vers q débutant et terminant loin des axes.

Notre stratégie de preuve qui consiste à étudier trois types d'exécutions partage des similarités avec l'approche de Leroux & Sutre [LS04]. Toutefois, Leroux & Sutre n'analysent pas la taille des schémas linéaires qu'ils obtiennent. Les schémas linéaires obtenus dans [LS04] sont parfois de taille au moins exponentielle en $|Q|$, parfois sans bornes apparentes. Cela ne pose pas problème dans [LS04] puisque le but y est de montrer que les 2-VASS sont aplatissables. Cependant, afin d'établir la complexité du problème d'accessibilité, il nous est crucial d'établir des « petites » bornes sur la taille des schémas linéaires et leur nombre de cycles. Cela s'avère

plus complexe et ne nous permet pas de suivre entièrement la démarche de Leroux & Sutre. De plus, bien que les schémas linéaires, que nous obtiendrons pour les exécutions demeurant près des axes, proviennent de l'aplatissement des 1-VASS, comme dans [LS04], l'argument utilisé dans [LS04] est incomplet puisqu'il dépend d'une preuve erronée, alors que nous exploitons ici les résultats de la section 4.3.

4.4.1 Accessibilité dans $\mathbb{N}^2$ près des axes

Dans cette section, nous montrons comment construire des schémas linéaires qui capturent les exécutions qui demeurent près des axes, tel qu'illustré dans le coin supérieur gauche de la figure 4.4. Intuitivement, les exécutions qui demeurent près des axes sont similaires aux exécutions d'un 1-VASS puisque l'une des composantes est bornée et peut donc être encodée dans les états de contrôle, au coût d'une explosion du nombre d'états de contrôle. Cette observation nous permet d'exploiter les résultats de la section 4.3.

Formellement, nous montrons le résultat suivant.

Proposition 4.6. *Soient $\mathcal{V} = (Q, T)$ un 2-VASS, $D \in \mathbb{N}$ et $\mathbb{L} = ([0, D] \times \mathbb{N}) \cup (\mathbb{N} \times [0, D])$. Il existe un ensemble fini $Y_{\mathbb{L}}$ de schémas linéaires tels que pour tous $p(\boldsymbol{u}), q(\boldsymbol{v}) \in \mathbb{L}$,*

(i) $p(\boldsymbol{u}) \to_{\mathbb{L}}^{*} q(\boldsymbol{v}) \implies p(\boldsymbol{u}) \xrightarrow{Y_{\mathbb{L}}}_{\mathbb{N}^2} q(\boldsymbol{v})$,

(ii) $|\rho| \leq (|Q| + \|T\| + D)^{O(1)}$ *et* $|\rho|_* \leq 2$ *pour tout* $\rho \in Y_{\mathbb{L}}$.

Avant de prouver la proposition 4.6, nous montrons d'abord comment capturer les exécutions qui demeurent près d'*un seul* axe, tel qu'illustré à la figure 4.5. Nous pourrons par la suite combiner les schémas linéaires obtenus afin de capturer les exécutions qui longent l'un *ou* l'autre des deux axes.

Lemme 4.7. *Soient $\mathcal{V} = (Q, T)$ un 2-VASS, $D \in \mathbb{N}$ et $\mathbb{B} \in \{([0, D] \times \mathbb{N}), (\mathbb{N} \times [0, D])\}$. Il existe un ensemble fini $Y_{\mathbb{B}}$ de schémas linéaires tel que pour tous $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{B}$,*

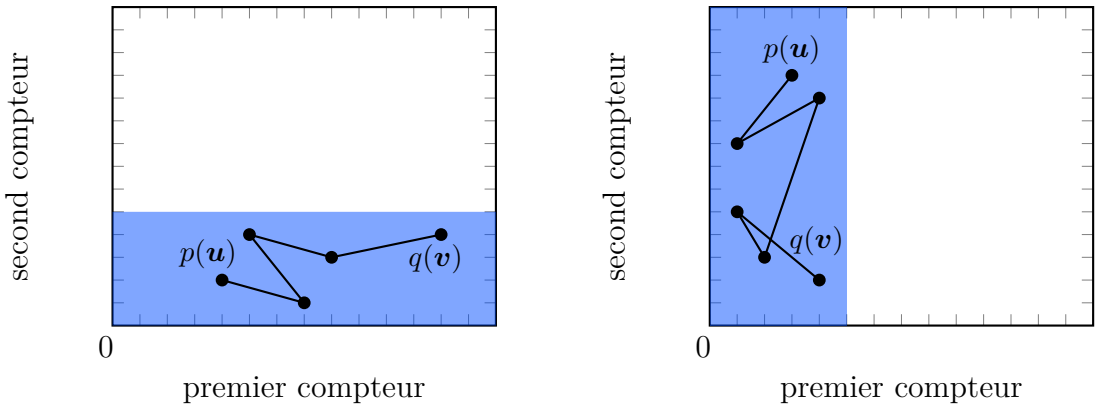


Figure 4.5 – Exemples d'exécutions qui demeurent près d'un seul axe dans $\mathbb{N}^2$.

(i) $p(\boldsymbol{u}) \to^*_{\mathbb{B}} q(\boldsymbol{v}) \iff p(\boldsymbol{u}) \xrightarrow{Y_{\mathbb{B}}}_{\mathbb{B}} q(\boldsymbol{v})$,

(ii) $|\rho| \leq (|Q| + \|T\| + D)^{O(1)}$ *et* $|\rho|_* \leq 1$ *pour tout* $\rho \in Y_{\mathbb{B}}$.

Démonstration. Nous considérons seulement le cas où $\mathbb{B} = \mathbb{N} \times [0, D]$, puisque l'autre cas est symétrique. Nous construisons un 1-VASS $\overline{\mathcal{V}} = (\overline{Q}, \overline{T})$, à partir de $\mathcal{V}$, tel que

(1) $\overline{Q} = \{q_i : q \in Q, i \in [0, D]\}$, et

(2) $p(u_1, u_2) \to^*_{\mathbb{B}} q(v_1, v_2)$ dans $\mathcal{V}$ si, et seulement si, $p_{u_2}(u_1) \to^*_{\mathbb{N}} q_{v_2}(v_1)$ dans $\overline{\mathcal{V}}$, pour tout $p(u_1, u_2), q(v_1, v_2) \in Q \times \mathbb{B}$.

Afin de satisfaire le point (2), il suffit de définir $\overline{T}$ de la façon suivante,

$$\overline{T} \stackrel{\text{déf}}{=} \{(p_n, i, q_{n+j}) : (p, (i, j), q) \in T \text{ et } n, n + j \in [0, D]\} \ .$$

Autrement dit, $\overline{\mathcal{V}}$ exploite le fait que le deuxième compteur de $\mathcal{V}$ est borné par D afin de l'encoder dans ses états de contrôle. Nous définissons le morphisme $\phi : \overline{T}^* \to T^*$ tel que $\phi(p_n, i, q_{n+j}) \stackrel{\text{déf}}{=} (p, (i, j), q)$ pour tout $(p_n, i, q_{n+j}) \in \overline{T}$. Pour tout $\overline{\pi} \in \mathsf{Chemins}_{\overline{\mathcal{V}}}(p, q)$,

$$\text{si } p_{u_2}(u_1) \xrightarrow{\overline{\pi}}_{\mathbb{N}} q_{v_2}(v_1) \text{ dans } \overline{\mathcal{V}}, \text{ alors } p(u_1, u_2) \xrightarrow{\phi(\overline{\pi})}_{\mathbb{B}} q(v_1, v_2) \text{ dans } \mathcal{V} \ . \qquad (4.1)$$

De plus, tout schéma linéaire $\overline{\rho} = \alpha_0\beta_1^*\alpha_1 \cdots \beta_k^*\alpha_k$ de $\overline{\mathcal{V}}$ induit un schéma linéaire $\phi(\overline{\rho}) = \phi(\alpha_0)\phi(\beta_1)^*\phi(\alpha_1) \cdots \phi(\beta_k)^*\phi(\alpha_k)$ de $\mathcal{V}$. Nous pouvons ainsi étendre ϕ naturellement aux ensembles de schémas linéaires $\overline{S}$,

$$\phi(\overline{S}) \stackrel{\text{déf}}{=} \{\phi(\overline{\rho}) : \overline{\rho} \in \overline{S}\} \ .$$

En appliquant la proposition 4.3 à $\overline{\mathcal{V}}$, nous obtenons un ensemble fini $\overline{Y}$ de schémas linéaires tel que

(a) $p(u) \to_{\mathbb{N}}^* q(v)$ dans $\overline{\mathcal{V}}$ si, et seulement si, $p(u) \xrightarrow{\overline{Y}}_{\mathbb{N}} q(v)$ dans $\overline{\mathcal{V}}$,

(b) pour tout $\overline{\rho} \in \overline{Y}$, $|\overline{\rho}| \leq (|\overline{Q}|+\|\overline{T}\|)^{O(1)} \leq (|Q|\cdot D+\|T\|)^{O(1)} = (|Q|+\|T\|+D)^{O(1)}$ et $|\overline{\rho}|_* \leq 1$.

Nous définissons l'ensemble $Y_{\mathbb{B}}$ de schémas linéaires, requis par le lemme, par $Y_{\mathbb{B}} \stackrel{\text{déf}}{=} \phi(\overline{Y})$. Par (b), $Y_{\mathbb{B}}$ satisfait le point (ii) du lemme. Nous concluons la preuve en montrant que $Y_{\mathbb{B}}$ satisfait également le point (i). Soient $p(u_1, u_2), q(v_1, v_2) \in Q \times \mathbb{B}$, nous avons la chaîne d'implications circulaire suivante,

$$\begin{aligned}
p(u_1,u_2) \to_{\mathbb{B}}^* q(v_1,v_2) \text{ dans } \mathcal{V} &\stackrel{(2)}{\iff} p_{u_2}(u_1) \longrightarrow_{\mathbb{N}}^* q_{v_2}(v_1) \quad \text{dans } \overline{\mathcal{V}} \\
&\stackrel{(a)}{\iff} p_{u_2}(u_1) \xrightarrow{\overline{Y}}_{\mathbb{N}} q_{v_2}(v_1) \quad \text{dans } \overline{\mathcal{V}} \\
&\stackrel{(4.1)}{\implies} p(u_1,u_2) \xrightarrow{\phi(\overline{Y})}_{\mathbb{B}} q(v_1,v_2) \quad \text{dans } \mathcal{V} \\
&\iff p(u_1,u_2) \xrightarrow{Y_{\mathbb{B}}}_{\mathbb{B}} q(v_1,v_2) \quad \text{dans } \mathcal{V} \\
&\implies p(u_1,u_2) \longrightarrow_{\mathbb{B}}^* q(v_1,v_2) \quad \text{dans } \mathcal{V}. \quad \square
\end{aligned}$$

Nous utilisons le lemme précédent afin de démontrer la proposition 4.6 qui est la proposition principale de cette sous-section.

Preuve de la proposition 4.6. Soient $\mathcal{V} = (Q, T)$ un 2-VASS et $\mathbb{D} \in \mathbb{N}$. Posons $\mathbb{L} = ([0, D] \times \mathbb{N}) \cup (\mathbb{N} \times [0, D])$.

Soient $E \stackrel{\text{déf}}{=} D + \|T\|$ et $\mathbb{L}' \stackrel{\text{déf}}{=} ([0,E]\times\mathbb{N}) \cup (\mathbb{N}\times[0,E])$. Soient $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q\times\mathbb{L}$ et $\pi \in \mathsf{Chemins}_{\mathcal{V}}(p,q)$ un chemin minimal tel que $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{L}} q(\boldsymbol{v})$. Afin de prouver la proposition 4.6, il suffit d'exhiber un schéma linéaire ρ tel que $p(\boldsymbol{u}) \xrightarrow{\rho}_{\mathbb{L}'} q(\boldsymbol{v})$, $|\rho| \leq (|Q| + \|T\| + D)^{O(1)}$ et $|\rho|_* \leq 2$. Soient $\mathbb{B}_1 \stackrel{\text{déf}}{=} [0,E]\times\mathbb{N}$, $\mathbb{B}_2 \stackrel{\text{déf}}{=} \mathbb{N}\times[0,E]$ et $\mathbb{H} \stackrel{\text{déf}}{=} \mathbb{B}_1 \cap \mathbb{B}_2 = [0,E]\times[0,E]$. Par minimalité de π et par définition de $\mathbb{H}$, nous remarquons que π entre dans la région $\mathbb{H}$ au plus $|\mathbb{H}|$ fois. Plus formellement, nous pouvons factoriser π sous la forme $\pi = \pi_1\pi_2\cdots\pi_k$, où

$$p_0(\boldsymbol{u}_0) \xrightarrow{\pi_1}_{\mathbb{L}} p_1(\boldsymbol{u}_1) \xrightarrow{\pi_2}_{\mathbb{L}} \cdots \xrightarrow{\pi_k}_{\mathbb{L}} p_k(\boldsymbol{u}_k)$$

et

(a) $p_0(\boldsymbol{u}_0) = p(\boldsymbol{u})$ et $p_k(\boldsymbol{u}_k) = q(\boldsymbol{v})$,

(b) $p_{i-1}(\boldsymbol{u}_{i-1}) \xrightarrow{\pi_i}_{\mathbb{C}_i} p_i(\boldsymbol{u}_i)$, où $\mathbb{C}_i \in \{\mathbb{B}_1, \mathbb{B}_2\}$ pour tout $i \in [k]$,

(c) $\boldsymbol{u}_i \in \mathbb{H}$ pour tout $i \in [k-1]$, et

(d) $k \leq |\mathbb{H}| = (E+1)^2 \leq D^{O(1)}$.

En combinant le point (b) et le lemme 4.7, nous observons que pour tout $i \in [k]$, il existe un schéma linéaire ρ_i tel que $p_{i-1}(\boldsymbol{u}_{i-1}) \xrightarrow{\rho_i}_{\mathbb{C}_i} p_i(\boldsymbol{u}_i)$, $|\rho_i| \leq (|Q| + \|T\| + E)^{O(1)} = (|Q| + \|T\| + D)^{O(1)}$ et $|\rho_i|_* \leq 1$.

Pour tout $i \in [k]$, nous écrivons $\rho_i = \alpha_i(\beta_i)^*\gamma_i$ où β_i est ou bien le cycle de ρ_i ou bien ε si $|\rho|_* = 0$. Soit $i \in [2, k-1]$. Puisque $|\rho_i| \leq (|Q| + \|T\| + D)^{O(1)}$ et $\boldsymbol{u}_{i-1}, \boldsymbol{u}_i \in \mathbb{H}$, il existe $e_i \leq (|Q| + \|T\| + E)^{O(1)}$ tel que $p_{i-1}(\boldsymbol{u}_{i-1}) \xrightarrow{\alpha_i(\beta_i)^{e_i}\gamma_i}_{\mathbb{C}_i} p_i(\boldsymbol{u}_i)$, et en particulier tel que

$$p_{i-1}(\boldsymbol{u}_{i-1}) \xrightarrow{\alpha_i(\beta_i)^{e_i}\gamma_i}_{\mathbb{L}'} p_i(\boldsymbol{u}_i) \ .$$

Ainsi, nous avons

$$p_0(\boldsymbol{u}_0) \xrightarrow{\alpha_1\beta_1^*\gamma_1}_{\mathbb{C}_1} p_1(\boldsymbol{u}_1) \xrightarrow{\prod_{i=2}^{k-1}\alpha_i\beta_i^{e_i}\gamma_i}_{\mathbb{L}'} p_{k-1}(\boldsymbol{u}_{k-1}) \xrightarrow{\alpha_k\beta_k^*\gamma_k}_{\mathbb{C}_k} p_k(\boldsymbol{u}_k) \ .$$

Afin de conclure la preuve, nous définissons le schéma linéaire

$$\rho \stackrel{\text{déf}}{=} \alpha_1\beta_1^*\gamma_1 \cdot \left(\prod_{i=2}^{k-1} \alpha_i\beta_i^{e_i}\gamma_i\right) \cdot \alpha_k\beta_k^*\gamma_k \ .$$

Nous remarquons que $p(\boldsymbol{u}) \xrightarrow{\rho}_{\mathbb{L}'} q(\boldsymbol{v})$, $|\rho|_* \leq 2$ et

$$\begin{aligned} |\rho| &\leq k \cdot \max\{e_i : i \in [2, k-1]\} \cdot \max\{|\rho_i| : i \in [1, k]\} \\ &\stackrel{(d)}{\leq} D^{O(1)} \cdot (|Q| + \|T\| + E)^{O(1)} \cdot (|Q| + \|T\| + D)^{O(1)} \\ &= (|Q| + \|T\| + D)^{O(1)}. \end{aligned}$$

□

4.4.2 Accessibilité dans $\mathbb{Z}^d$

Avant d'analyser les exécutions demeurant loin des axes, nous devons d'abord étudier l'accessibilité sans restrictions, c'est-à-dire dans $\mathbb{Z}^2$ plutôt que $\mathbb{N}^2$. Nous montrerons que pour tout 2-VASS, la relation $\to_{\mathbb{Z}^2}$ est capturée dans $\mathbb{Z}^2$ par un ensemble fini de schémas linéaires, chacun de taille polynomiale. Par souci de généralité, nous ne nous restreignons pas à deux compteurs et prouvons ce résultat dans sa pleine généralité :

Proposition 4.8. *Soit $\mathcal{V} = (Q, T)$ un d-VASS. Il existe un ensemble fini S de schémas linéaires tel que pour tous $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{Z}^d$,*

(i) $p(\boldsymbol{u}) \to^*_{\mathbb{Z}^d} q(\boldsymbol{v}) \iff p(\boldsymbol{u}) \xrightarrow{S}_{\mathbb{Z}^d} q(\boldsymbol{v})$,

(ii) $|\rho| \leq 2 \cdot |Q| \cdot |T|$ *et* $|\rho|_* \leq |T|$ *pour chaque* $\rho \in S$.

Afin de prouver la proposition 4.8, nous démontrons que les images de Parikh dans un graphe dirigé et étiqueté, ou de façon équivalente dans les automates non déterministes, peuvent être représentées par une union finie de schémas linéaires chacun de taille polynomiale. Bien que ce type de résultat existe dans la littérature (p. ex. dans [LS04, lemme 4.1] et [To10, proposition 7.3.4]), il ne semble exister aucun résultat qui borne de telles représentations polynomialement.

Lemme 4.9. *Soit $G = (U, E)$ un graphe dirigé fini dont les arcs sont étiquetés d'éléments d'un alphabet fini Σ. Il existe un ensemble fini S de schémas linéaires tel que*

(i) $\mathsf{Parikh}(\mathsf{Chemins}_G) = \bigcup_{\rho \in S} \mathsf{Parikh}(\rho)$,

(ii) $|\rho| \leq 2 \cdot |U| \cdot |E|$ *et* $|\rho|_* \leq |E|$ *pour tout* $\rho \in S$.

La preuve du lemme 4.9 étant quelque peu technique, nous introduisons quelques définitions et nous procédons en deux parties.

Soit $G = (U, E)$ un graphe fini dirigé et étiqueté par un alphabet fini Σ. Pour chaque $u \in U$, nous définissons $\mathsf{entrant}(u) \stackrel{\text{déf}}{=} \{(u', a, u'') \in E : u'' = u\}$ et $\mathsf{sortant}(u) \stackrel{\text{déf}}{=} \{(u', a, u'') \in E : u' = u\}$ respectivement l'ensemble des arcs entrants et celui des arcs sortants de u. Soit $\boldsymbol{\sigma} \in \mathbb{N}^E$, nous disons que $\boldsymbol{\sigma}$ *conserve le flot* si pour tout $u \in U$ nous avons

$$\sum_{e \in \mathsf{entrant}(u)} \boldsymbol{\sigma}(e) \;=\; \sum_{e \in \mathsf{sortant}(u)} \boldsymbol{\sigma}(e) \;.$$

Proposition 4.10. *Soit $G = (U, E)$ un graphe dirigé fini et étiqueté d'un alphabet fini Σ. Soit $\pi \in \mathsf{Chemins}_G$. Il existe $h \in \mathbb{N}_{>0}$, des schémas linéaires $\rho_1, \rho_2, \ldots, \rho_h \subseteq E^*$, et $\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \ldots, \boldsymbol{\sigma}_h \in \mathbb{N}^E$ tels que*

(a) $|\rho_1| \leq |U| \cdot |E|$, $|\rho_1|_* = 0$, *et ρ_1 visite chaque sommet de π au moins une fois,*

(b) $\boldsymbol{\sigma}_1$ *conserve le flot, et*

(c) $\mathsf{Parikh}(\pi) = \mathsf{Parikh}(\rho_1) + \boldsymbol{\sigma}_1$,

et pour tout $1 < i \leq h$,

(1) ρ_i *est un schéma linéaire obtenu à partir de ρ_1 en insérant $i - 1$ cycles simples de la forme β^*,*

(2) $\boldsymbol{\sigma}_i$ *conserve le flot,*

(3) $\mathsf{Parikh}(\rho_{i-1}) + \boldsymbol{\sigma}_{i-1} \subseteq \mathsf{Parikh}(\rho_i) + \boldsymbol{\sigma}_i$,

(4) $\boldsymbol{\sigma}_{i-1}(e) \geq \boldsymbol{\sigma}_i(e)$ *pour tout* $e \in E$ *et il existe* $e \in E$ *t.q.* $\boldsymbol{\sigma}_{i-1}(e) > \boldsymbol{\sigma}_i(e) = 0$*, et*

(5) $\boldsymbol{\sigma}_h = \mathbf{0}$.

Démonstration. Soit $\pi \in \mathsf{Chemins}_G$. Nous construisons ρ_1 et $\boldsymbol{\sigma}_1$ tels que les points (a), (b) et (c) sont satisfaits. Le chemin π peut être décomposé sous la forme $\pi = e_1\pi_1 \cdots e_k\pi_k$ où $k \leq |U|$ et chaque $e_j = (u, a, u')$ est la première transition telle que u ou u' apparaît dans π. Nous construisons ρ_1 et $\boldsymbol{\sigma}_1$ itérativement de la façon suivante : nous posons initialement $\rho_1 = \pi$ et $\boldsymbol{\sigma}_1 = \mathbf{0}$; nous retirons ensuite successivement un cycle simple β de l'un des π_j et ajoutons $\mathsf{Parikh}(\beta)$ à $\boldsymbol{\sigma}_1$; nous répétons ce processus, avec potentiellement d'autres cycles, jusqu'à ce qu'il ne soit plus possible de continuer. Le schéma linéaire ρ_1 résultant est un chemin de longueur au plus $|U| \cdot |E|$. De plus, $\boldsymbol{\sigma}_1$ conserve le flot, puisqu'il est la somme d'images de cycles, et par construction, $\mathsf{Parikh}(\pi) = \mathsf{Parikh}(\rho_1) + \boldsymbol{\sigma}_1$. Ainsi les points (a), (b) et (c) sont satisfaits.

Montrons les points (1) à (5) par induction sur $1 < i \leq h$. Nous démontrons seulement le cas de base, l'étape d'induction pouvant être prouvée similairement. Soit $E' \stackrel{\text{déf}}{=} \{e \in E : \boldsymbol{\sigma}_{i-1}(e) > 0\}$. Si $E' = \emptyset$, il suffit de prendre $h = 1$ et nous avons ainsi terminé puisque le point (5) est satisfait. Supposons donc que $E' \neq \emptyset$. Considérons une fonction $\chi : E' \to E'$ qui satisfait

$$\chi(u_1, a, u_2) = (u_1', a, u_2') \implies u_2 = u_1'.$$

Notons que χ existe puisque $\boldsymbol{\sigma}_{i-1}$ conserve le flot. Par le principe du pigeonnier, il existe $e \in E'$ et $\ell \geq 0$ tels que

$$\beta \stackrel{\text{déf}}{=} e\chi(e)\chi^2(e) \cdots \chi^\ell(e)$$

est un cycle simple. Sans perte de généralité, nous pouvons supposer que $c \stackrel{\text{déf}}{=} \boldsymbol{\sigma}_{i-1}(e) = \min\{\boldsymbol{\sigma}_{i-1}(\chi^h(e)) : 0 \leq h \leq \ell\}$, ainsi $\boldsymbol{\sigma}_{i-1}(e)$ est minimal parmi les arcs présents sur le cycle simple β.

Nous définissons $\boldsymbol{\sigma}_i \stackrel{\text{déf}}{=} \boldsymbol{\sigma}_{i-1} - \mathsf{Parikh}(\beta^c)$ et notons que $\boldsymbol{\sigma}_i$ conserve le flot puisque

β est un cycle, et $\boldsymbol{\sigma}_i \in \mathbb{N}^E$ par minimalité de c ; ainsi les points (2) et (4) sont satisfaits. Soit u le premier sommet de l'arc e, alors β est un cycle simple de u vers u. Par (1), le schéma linéaire ρ_{i-1} peut être obtenu à partir de ρ_1 en insérant $(i-2)$ cycles simples, et peut ainsi être factorisé sous $\rho_{i-1} = \alpha\gamma$, où α est un schéma linéaire terminant en u. Nous posons $\rho_i \stackrel{\text{déf}}{=} \alpha\beta^*\gamma$ et ainsi le point (1) est satisfait. De plus, le point (3) est satisfait car $\mathsf{Parikh}(\rho_{i-1}) + \boldsymbol{\sigma}_{i-1} = \mathsf{Parikh}(\rho_{i-1}) + \mathsf{Parikh}(\beta^c) + \boldsymbol{\sigma}_i \subseteq \mathsf{Parikh}(\rho_i) + \boldsymbol{\sigma}_i$. □

Nous sommes maintenant en mesure de prouver le lemme 4.9.

Preuve du lemme 4.9. Nous posons

$$S \stackrel{\text{déf}}{=} \{\rho : \rho \text{ est un schéma linéaire}, |\rho| \leq 2 \cdot |U| \cdot |E| \text{ et } |\rho|_* \leq |E|\} \ .$$

Trivialement, le point (ii) est satisfait. Démontrons le point (i). Soit $\pi \in \mathsf{Chemins}_G$. Soient les schémas linéaires $\rho_1, \rho_2, \ldots, \rho_h \subseteq E^*$, et les vecteurs $\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \ldots, \boldsymbol{\sigma}_h \in \mathbb{N}^E$ obtenus pour π par la proposition 4.10. Nous avons

$$\mathsf{Parikh}(\pi) \stackrel{(c)}{=} \mathsf{Parikh}(\rho_1) + \boldsymbol{\sigma}_1 \stackrel{(3)}{\subseteq} \mathsf{Parikh}(\rho_2) + \boldsymbol{\sigma}_2 \stackrel{(3)}{\subseteq} \cdots \stackrel{(3)}{\subseteq} \mathsf{Parikh}(\rho_h) + \boldsymbol{\sigma}_h \stackrel{(5)}{=} \mathsf{Parikh}(\rho_h) \ .$$

Ainsi $\mathsf{Parikh}(\pi) \subseteq \mathsf{Parikh}(\rho_h)$, il suffit donc de borner la taille de ρ_h. Par le point (4) de la proposition 4.10, nous avons $h \leq |E|$, et par le point (1) nous avons $|\rho_i| \leq |\rho_1| + |U| \cdot (i-1)$. Ainsi, $|\rho_h| \leq |\rho_1| + |U| \cdot |E| \leq 2 \cdot |U| \cdot |E|$, où la derniere inégalité découle du point (a). De plus, $|\rho_h|_* \leq |E|$ par le point (1) et $h \leq |E|$. □

Nous démontrons maintenant la proposition 4.8 qui est la proposition principale de cette sous-section et qui démontre que $\rightarrow^*_{\mathbb{Z}^d}$ est plate pour $\mathbb{Z}^d$.

Preuve de la proposition 4.8. Soit $\mathcal{V} = (Q, T)$ un d-VASS. Nous avons $T \subseteq Q \times \Sigma \times Q$ pour un sous-ensemble fini $\Sigma \subseteq \mathbb{Z}^d$. Soit S l'ensemble fini de schémas linéaires obtenus par le lemme 4.9, alors le point (ii) est satisfait. Le point (i) est également satisfait car,

$$
\begin{aligned}
p(\boldsymbol{u}) \to^*_{\mathbb{Z}^d} q(\boldsymbol{v}) \quad &\iff \quad \exists \pi \in \mathsf{Chemins}_{\mathcal{V}}(p,q) \text{ t.q. } \boldsymbol{v} - \boldsymbol{u} = \sum_{\boldsymbol{z} \in \Sigma} \mathsf{Parikh}(\pi)(\boldsymbol{z}) \cdot \boldsymbol{z} \\
&\overset{\text{lemme 4.9 (i)}}{\iff} \quad \exists \rho \in S(p,q),\ \exists \boldsymbol{x} \in \mathsf{Parikh}(\rho) \text{ t.q. } \boldsymbol{v} - \boldsymbol{u} \in \sum_{\boldsymbol{z} \in \Sigma} \boldsymbol{x}(\boldsymbol{z}) \cdot \boldsymbol{z} \\
&\iff \quad \exists \rho \in S(p,q) \text{ t.q. } \boldsymbol{v} - \boldsymbol{u} \in \delta(\rho) \\
&\iff \quad p(\boldsymbol{u}) \xrightarrow{S}_{\mathbb{Z}^d} q(\boldsymbol{v}) \ . \qquad \square
\end{aligned}
$$

4.4.3 Accessibilité dans $\mathbb{N}^2$ loin des axes

Le but de cette sous-section est de montrer que pour tout 2-VASS $\mathcal{V}$, il existe une « petite » borne $D \in \mathbb{N}$ telle que la relation $\to^*_{[D,\infty)^2}$ est plate. Dans [LS04], un d-VASS qui respecte cette propriété est dit *ultimement aplatissable*[4]. Intuitivement, nous sommes intéressés par ce type d'exécutions car la difficulté de l'analyse des exécutions restreintes à $\mathbb{N}^2$ apparaît « près » des axes, or, lorsqu'une exécution a lieu « loin » des deux axes, elle se comporte essentiellement comme une exécution dans $\mathbb{Z}^2$. Ainsi, nous pourrons exploiter le fait que $\to^*_{\mathbb{Z}^2}$ est plate pour $\mathbb{Z}^2$.

Afin de démontrer ce résultat, nous analysons dans un premier temps les exécutions restreintes à $\mathbb{N}^2$ qui débutent et se terminent dans un même état de contrôle, et dans $[D,\infty)^2$ sans nécessairement y demeurer (voir au bas de la figure 4.4).

Formellement, le résultat principal de cette sous-section est le suivant :

Proposition 4.11. *Soit $\mathcal{V} = (Q,T)$ un 2-VASS. Il existe $D \leq (|Q| + \|T\|)^{O(1)}$ et des ensembles finis de schémas linéaires R, X tels que pour tous $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{O}$, où $\mathbb{O} \overset{\text{déf}}{=} [D,\infty)^2$,*

(a)
- $q(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v}) \iff q(\boldsymbol{u}) \xrightarrow{R}_{\mathbb{N}^2} q(\boldsymbol{v})$,
- $|\rho| \leq (|Q| + \|T\|)^{O(1)}$ *et* $|\rho|_* \leq 2$ *pour tout* $\rho \in R$.

(b)
- $p(\boldsymbol{u}) \to^*_{\mathbb{O}} q(\boldsymbol{v}) \implies p(\boldsymbol{u}) \xrightarrow{X}_{\mathbb{N}^2} q(\boldsymbol{v})$,

[4] Ou, plus littéralement, *ultimement plat* de l'anglais « ultimately flat ».

- $|\rho| \leq (|Q| + \|T\|)^{O(1)}$ *et* $|\rho|_* \leq 2 \cdot |Q|$ *pour tout* $\rho \in X$.

La preuve de cette proposition requiert deux étapes intermédiaires. Premièrement, au lemme 4.12, nous montrons qu'étant donné un vecteur $\boldsymbol{b}$ d'un ensemble fini $P \subseteq \mathbb{Z}^2$, l'ensemble des vecteurs de $L = \mathsf{Lin}(\boldsymbol{b}, P)$ appartenant à un quadrant Z est égal à un ensemble semi-linéaire $\bigcup_i \mathsf{Lin}(\boldsymbol{c}_i, P_i)$ où chaque vecteur de P_i est dans le quadrant Z et provient de $L \cup P$. Des décompositions similaires forment la pierre angulaire des résultats de Hopcroft & Pansiot [HP79] et de Leroux & Sutre [LS04]. La contribution du lemme 4.12 consiste à établir une nouvelle preuve à partir de laquelle il est possible d'obtenir des bornes sur la taille de ces ensembles semi-linéaires, c.-à-d. obtenir des c_i et P_i de « petite » norme.

Par la suite, au lemme 4.13 nous montrons comment exploiter ces décompositions afin de construire des schémas linéaires possédant au plus deux cycles pointant dans le même quadrant. Cela nous permettra ensuite de prouver le point (a) de la proposition 4.11, à partir duquel nous prouverons le point (b).

Lemme 4.12. *Soient* $P \subseteq \mathbb{Z}^2$ *et* $\boldsymbol{b} \in P$. *Pour tout quadrant* Z, *il existe* $e \leq (|P| \cdot \|P\|)^{O(1)}$, $m \in \mathbb{N}_{>0}$ *et*

$$\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_m \in Z \cap \mathsf{Lin}_{[0,e]}(\boldsymbol{b}, P)$$
$$P_1, P_2, \ldots, P_m \subseteq Z \cap [\mathsf{Lin}_{[0,e]}(\boldsymbol{b}, P) \cup P]$$

tels que pour tout $i \in [m]$, $|P_i| \leq 2$, *et*

$$\mathsf{Lin}(\boldsymbol{b}, P) \cap Z = \bigcup_{i \in [m]} \mathsf{Lin}(\boldsymbol{c}_i, P_i) \ .$$

Afin de faciliter la lecture de cette section, la preuve du lemme 4.12 est présentée à l'annexe A puisque celle-ci est particulièrement technique.

Nous donnons une explication intuitive de la façon dont nous pouvons prouver la proposition 4.11 (a) grâce au lemme 4.12. Considérons une exécution d'un 2-VASS $\mathcal{V}$ débutant en $q(\boldsymbol{u})$ et terminant en $q(\boldsymbol{v})$. Par la proposition 4.8, nous savons que $\to^*_{\mathbb{Z}^2}$ est capturée par un ensemble fini de schémas linéaires. Puisque l'exécution débute

et termine dans le même état, il est possible de montrer que ces schémas linéaires correspondent essentiellement à un ensemble linéaire $\mathsf{Lin}(\boldsymbol{b}, P)$ tel que $\boldsymbol{b} \in P$. Par le lemme 4.12, il est possible de décomposer cet ensemble en un ensemble semi-linéaire dont toutes les périodes pointent dans le même quadrant. Le point crucial est que chaque ensemble linéaire de cet ensemble semi-linéaire peut être reconverti en un schéma linéaire ρ tel que $|\rho|_* \leq 2$ et tel que les cycles de ρ pointent dans le même quadrant. Ainsi, informellement, tout chemin d'un tel schéma linéaire ne « dérive pas trop » et, si $\boldsymbol{u}$ et $\boldsymbol{v}$ sont suffisamment grands, $\rightarrow^*_{\mathbb{Z}^2}$ et $\rightarrow^*_{\mathbb{N}^2}$ « coïncident ». Pour illustrer cette idée, considérons un schéma linéaire $\rho = \alpha_0\beta_1^*\alpha_1\beta_2^*\alpha_2$ tel que

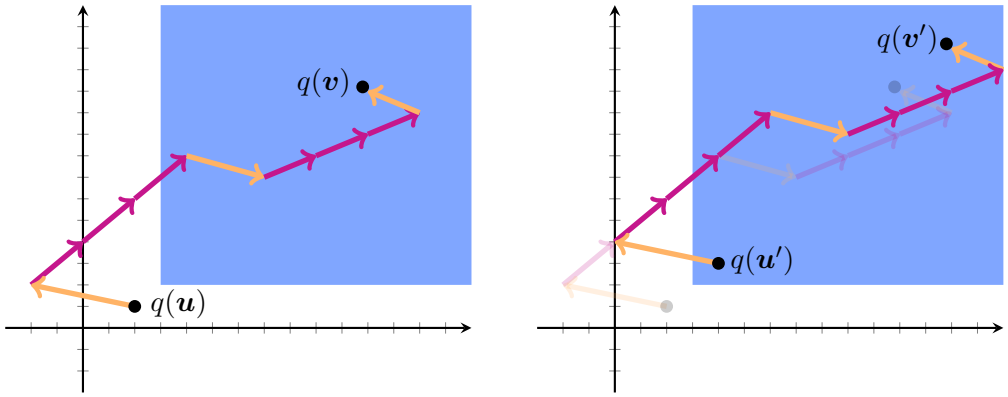


Figure 4.6 – Exemple d'exécution d'un schéma linéaire $\rho = \alpha_0\beta_1^*\alpha_1\beta_2^*\alpha_2$ sans zig-zag, c'est-à-dire, tel que tous les cycles de ρ pointent dans le même quadrant ; en l'occurrence $\mathbb{N} \times \mathbb{N}$. Les flèches colorées ▬ correspondent aux déplacements de α_0, α_1 et α_2, et les flèches colorées en ▬ correspondent aux déplacements de β_1 et β_2.

$\delta(\beta_1), \delta(\beta_2) \in \mathbb{N}^2$. Soient $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{N}^2$, supposons que $q(\boldsymbol{u}) \xrightarrow{\rho}_{\mathbb{Z}^2} q(\boldsymbol{v})$. Nous illustrons une telle exécution du côté gauche de la figure 4.6. L'exécution qui mène de $q(\boldsymbol{u})$ à $q(\boldsymbol{v})$ ne peut chuter très bas sous zéro puisque seuls α_0, α_1 et α_2 peuvent contribuer négativement au déplacement de l'exécution. Ainsi, si $\boldsymbol{u}$ et $\boldsymbol{v}$ sont suffisamment grands, l'exécution est également une exécution restreinte à $\mathbb{N}^2$. Nous illustrons cette observation du côté droit de la figure 4.6.

Ainsi, nous montrons maintenant comment appliquer le lemme 4.12 aux schémas linéaires. Nous utilisons la terminologie de [LS04], et nous disons qu'un schéma

linéaire ρ est *sans zigzag* si $\mathsf{Cycles}(\rho) \subseteq Z$ pour un quadrant Z.

Lemme 4.13. *Soient $\mathcal{V} = (Q, T)$ un 2-VASS, et $q \in Q$. Pour tout schéma linéaire ρ de q vers q, il existe un ensemble fini R_ρ de schémas linéaires sans zigzag tel que*

(i) $\delta(\rho) \subseteq \delta(R_\rho)$,

(ii) $|\sigma| \leq (|\rho| + \|T\|)^{O(1)}$ *et* $|\sigma|_* \leq 2$ *pour tout* $\sigma \in R_\rho$.

Démonstration. Soit $\rho = \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$ un schéma linéaire de q vers q. Sans perte de généralité, nous supposons que $\delta(\alpha_0 \cdots \alpha_k) \in \mathsf{Cycles}(\rho)$, car autrement nous pouvons plutôt considérer le schéma linéaire $\rho' \stackrel{\text{déf}}{=} (\alpha_0 \cdots \alpha_k)^* \rho$ qui satisfait cette propriété et qui est tel que $\delta(\rho) \subseteq \delta(\rho')$. De plus, nous supposons que $\delta(\beta_i) \neq \delta(\beta_j)$ lorsque $i \neq j$, car autrement il suffit de retirer β_j de ρ, ce qui ne change pas l'ensemble des déplacements de ρ.

Nous pouvons écrire $\delta(\rho) = \bigcup_{\text{quadrant } Z} \delta(\rho) \cap Z$. Ainsi, pour chaque quadrant Z, il suffit de construire un ensemble de schémas linéaires sans zigzag $R_{\rho,Z}$ approprié tel que $\delta(\rho) \cap Z = \delta(R_{\rho,Z})$. La preuve est ainsi complétée en prenant $R_\rho \stackrel{\text{déf}}{=} \bigcup_Z R_{\rho,Z}$.

Soient $\boldsymbol{b} \stackrel{\text{déf}}{=} \delta(\alpha_0 \cdots \alpha_k)$ et $P \stackrel{\text{déf}}{=} \mathsf{Cycles}(\rho)$. Par hypothèse, nous avons $\boldsymbol{b} \in P$. Notons que $\|P\| \leq |\rho| \cdot \|T\|$. Par le lemme 4.12 il existe $e \leq (|P| \cdot \|P\|)^{O(1)}$, $m \in \mathbb{N}_{>0}$ et

$$\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_m \in Z \cap \mathsf{Lin}_{[0,e]}(\boldsymbol{b}, P)$$
$$P_1, P_2, \ldots, P_m \subseteq Z \cap [\mathsf{Lin}_{[0,e]}(\boldsymbol{b}, P) \cup P]$$

tels que pour tout $i \in [m]$, $|P_i| \leq 2$, et

$$\mathsf{Lin}(\boldsymbol{b}, P) \cap Z = \bigcup_{i \in [m]} \mathsf{Lin}(\boldsymbol{c}_i, P_i) \ .$$

Soit $i \in [m]$. Pour tout $\boldsymbol{u} \in \{\boldsymbol{c}_i\} \cup (P_i \cap \mathsf{Lin}(\boldsymbol{b}, P))$, il existe un chemin $\pi_{\boldsymbol{u}}$ de q vers q de la forme $\alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k$, pour certains $0 \leq e_1, e_2, \ldots, e_k \leq e \leq (|\rho|^2 \cdot \|T\|)^{O(1)}$, tel que $\boldsymbol{u} = \delta(\pi_{\boldsymbol{u}})$. Notons que $|\pi_{\boldsymbol{u}}| \leq (|\rho| + \|T\|)^{O(1)}$. Soit $\pi_{\boldsymbol{c}_i} = \alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k$ le tel chemin obtenu pour $\boldsymbol{c}_i$. Nous définissons le schéma linéaire

σ_i obtenu à partir de $\pi_{\boldsymbol{c}_i}$ en insérant "β_j^*", pour tout $j \in [k]$, lorsque $\delta(\beta_j) \in P_i \cap P$. Plus formellement, nous définissons $\sigma_i \stackrel{\text{déf}}{=} \alpha_0 \beta_1^{e_1} \theta_1 \alpha_1 \cdots \beta_k^{e_k} \theta_k \alpha_k$ où

$$\theta_j = \begin{cases} \beta_j^* & \text{si } \delta(\beta_j) \in P_i \cap P \\ \varepsilon & \text{sinon} \end{cases} \qquad \text{pour tout } j \in [k] \ .$$

Rappelons que $P_i \subseteq Z \cap [\mathsf{Lin}(\boldsymbol{b}, P) \cup P]$. Ainsi,

$$\rho_i \stackrel{\text{déf}}{=} \sigma_i \cdot \prod_{\boldsymbol{u} \in P_i \setminus P} \pi_{\boldsymbol{u}}^*$$

est un schéma linéaire sans zigzag tel que $|\rho_i|_* \leq 2$, $\mathsf{Cycles}(\rho_i) \subseteq Z$, et qui satisfait $\delta(\rho_i) = \mathsf{Lin}(\boldsymbol{c}_i, P_i)$ et $|\rho_i| \leq (|\rho| + \|T\|)^{O(1)}$. Nous concluons la preuve en posant $R_{\rho,Z} \stackrel{\text{déf}}{=} \bigcup_{i \in I} \rho_i$ et en remarquant que

$$\begin{aligned} \delta(\rho) \cap Z &= \mathsf{Lin}(\boldsymbol{b}, P) \cap Z \\ &\subseteq \bigcup_{i \in I} \mathsf{Lin}(\boldsymbol{c}_i, P_i) \\ &= \bigcup_{i \in I} \delta(\rho_i) \\ &= \delta(R_{\rho,Z}). \end{aligned}$$

□

La proposition suivante nous sera cruciale afin de démontrer la proposition 4.11. Nous présentons la preuve de [LS04] dans notre terminologie afin d'être complet.

Proposition 4.14 ([LS04, lemme 4.6])**.** *Soient $\mathcal{V} = (Q, T)$ un d-VASS et $\rho = \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$ un schéma linéaire sans zigzag. Pour tous $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^d$, si $p(\boldsymbol{u}) \xrightarrow{\rho}_{\mathbb{Z}^d} q(\boldsymbol{v})$ et $\|\boldsymbol{u}\|, \|\boldsymbol{v}\| \geq |\rho| \cdot \|T\|$, alors $p(\boldsymbol{u}) \xrightarrow{\rho}_{\mathbb{N}^d} q(\boldsymbol{v})$.*

Démonstration. Soient $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^d$ tels que $p(\boldsymbol{u}) \xrightarrow{\rho}_{\mathbb{Z}^d} q(\boldsymbol{v})$ et $\|\boldsymbol{u}\|, \|\boldsymbol{v}\| \geq |\rho| \cdot \|T\|$. Il existe $e_1, e_2, \ldots, e_k \in \mathbb{N}$ et $\pi = \alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k$ tels que $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{Z}^d} q(\boldsymbol{v})$. Il suffit de montrer que pour tous $\pi', \pi'' \in T^*$ tels que $\pi = \pi' \pi''$, nous avons

$\boldsymbol{u} + \delta(\pi') \geq \mathbf{0}$. Considérons une telle décomposition. Le préfixe π' peut découper π de deux façons possibles :

$$\pi' = \begin{cases} \alpha_0 \beta_1^{e_1} \cdots \alpha_{i-1} \beta_i^{e_i} \gamma & \gamma \text{ est un préfixe de } \alpha_i \text{ pour un certain } i \in [0, k] \\ \alpha_0 \beta_1^{e_1} \cdots \alpha_{i-1} \beta_i^{e'_i} \gamma & e'_i < e_i \text{ et } \gamma \text{ est un préfixe de } \beta_i \text{ pour un certain } i \in [k] \end{cases}$$

Nous ne considérons que le deuxième cas, le premier étant similaire. Soit $j \in [d]$. Puisque ρ est sans zigzag, $\delta(\beta_1)(j), \delta(\beta_2)(j), \ldots, \delta(\beta_k)(j)$ sont tous non négatifs, ou tous non positifs. Nous explorons les deux cas.

$\underline{\delta(\beta_1)(j), \delta(\beta_2)(j), \ldots, \delta(\beta_k)(j) \geq 0}$: Nous avons

$$\begin{aligned} \boldsymbol{u}(j) + \delta(\pi')(j) &= \boldsymbol{u}(j) + \delta(\alpha_0 \beta_1^{e_1} \cdots \alpha_{i-1} \beta_i^{e'_i} \gamma)(j) \\ &= \boldsymbol{u}(j) + \delta(\alpha_0 \alpha_1 \cdots \alpha_{i-1} \gamma)(j) + \sum_{\ell=1}^{i-1} e_\ell \cdot \delta(\beta_\ell)(j) + e'_i \cdot \delta(\beta_i)(j) \\ &\geq \boldsymbol{u}(j) + \delta(\alpha_0 \alpha_1 \cdots \alpha_{i-1} \gamma)(j) \\ &\geq \boldsymbol{u}(j) - |\rho| \cdot \|T\| \\ &\geq 0 \ . \end{aligned}$$

$\underline{\delta(\beta_1)(j), \delta(\beta_2)(j), \ldots, \delta(\beta_k)(j) \leq 0}$: Il existe $\gamma' \in T^*$ un suffixe de β_i tel que $\beta_i^{e_i} = \beta_i^{e'_i} \gamma \gamma' \beta_i^{e''_i}$ où $e''_i = e_i - e'_i - 1 \geq 0$. Ainsi,

$$\begin{aligned} \boldsymbol{u}(j) + \delta(\pi')(j) &= \boldsymbol{v}(j) - \delta(\pi'')(j) \\ &= \boldsymbol{v}(j) - \delta(\gamma' \beta_i^{e''_i} \alpha_i \beta_{i+1}^{e_{i+1}} \cdots \beta_k^{e_k} \alpha_k)(j) \\ &= \boldsymbol{v}(j) - \delta(\gamma' \alpha_i \cdots \alpha_k)(j) - e''_i \cdot \delta(\beta_i)(j) - \sum_{\ell=i+1}^{k} e_\ell \cdot \delta(\beta_\ell)(j) \\ &\geq \boldsymbol{v}(j) - \delta(\gamma' \alpha_i \cdots \alpha_k)(j) \\ &\geq \boldsymbol{v}(j) - |\rho| \cdot \|T\| \\ &\geq 0 \ . \end{aligned}$$

□

Nous sommes maintenant en mesure de prouver la proposition 4.11 qui est la proposition principale de cette sous-section.

Preuve de la proposition 4.11. Soit $\mathcal{V} = (Q, T)$ un 2-VASS.

(a) Soit S l'ensemble de schémas linéaires de la proposition 4.8 obtenu pour $\mathcal{V}$ et tel que pour tout $q(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{Z}^2$,

- $q(\boldsymbol{u}) \to^*_{\mathbb{Z}^d} q(\boldsymbol{v}) \iff q(\boldsymbol{u}) \xrightarrow{S}_{\mathbb{Z}^d} q(\boldsymbol{v})$,
- $|\rho| \leq 2 \cdot |Q| \cdot |T|$ et $|\rho|_* \leq |T|$ pour tout $\rho \in S$.

Soient $q \in Q$ et $\rho \in S(q, q)$, nous définissons $R_{q,\rho}$ comme étant l'ensemble de schémas linéaires sans zigzag obtenu par le lemme 4.13 pour q et ρ. Posons

$$R \overset{\text{déf}}{=} \bigcup_{q \in Q} \bigcup_{\rho \in S(q,q)} R_{q,\rho} \ .$$

Par le point (ii) du lemme 4.13 nous avons

$$|\sigma| \ \leq \ (2 \cdot |Q| \cdot |T| + \|T\|)^{O(1)} \ = \ (|Q| + \|T\|)^{O(1)}$$

pour tout $\sigma \in R$. Nous choisissons $D \in \mathbb{N}$ requis dans l'énoncé de la proposition comme étant $D \overset{\text{déf}}{=} \max\{|\sigma| : \sigma \in R\} \cdot \|T\| \leq (|Q| + \|T\|)^{O(1)}$, et posons $\mathbb{O} = [D, \infty]^2$. Afin de conclure la preuve du point (a), nous montrons que, pour tous $q(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{O}$, nous avons $q(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v}) \iff q(\boldsymbol{u}) \xrightarrow{\sigma}_{\mathbb{N}^2} q(\boldsymbol{v})$ pour un certain $\sigma \in R$:

$$\begin{aligned}
q(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v}) \quad &\Longrightarrow \quad q(\boldsymbol{u}) \to^*_{\mathbb{Z}^2} q(\boldsymbol{v}) \\
&\overset{\text{proposition 4.8}}{\Longleftrightarrow} \quad q(\boldsymbol{u}) \xrightarrow{\rho}_{\mathbb{Z}^2} q(\boldsymbol{v}) \text{ pour un certain } \rho \in S \\
&\overset{\text{lemme 4.13 (i)}}{\Longrightarrow} \quad q(\boldsymbol{u}) \xrightarrow{\sigma}_{\mathbb{Z}^2} q(\boldsymbol{v}) \text{ pour certains } \sigma \in R_\rho, \rho \in S \\
&\overset{\text{proposition 4.14}}{\Longrightarrow} \quad q(\boldsymbol{u}) \xrightarrow{\sigma}_{\mathbb{N}^2} q(\boldsymbol{v}) \text{ pour un certain } \sigma \in R \\
&\Longrightarrow \quad q(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v}) \ .
\end{aligned}$$

(b) Soient $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{O}$ tels que $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{O}} q(\boldsymbol{v})$ pour un certain $\pi \in \mathsf{Chemins}_{\mathcal{V}}(p, q)$. Il est possible de factoriser π sous la forme $\pi = \alpha_0\beta_1\alpha_1 \cdots \beta_k\alpha_k$ où $\alpha_0, \alpha_1, \ldots, \alpha_k \leq |Q|$ et $k \leq |Q|$, de telle sorte que

$$p(\boldsymbol{u}) \xrightarrow{\alpha_0}_{\mathbb{O}} q_1(\boldsymbol{u}_1) \xrightarrow{\beta_1}_{\mathbb{O}} q_1(\boldsymbol{u}'_1) \xrightarrow{\alpha_1}_{\mathbb{O}} q_2(\boldsymbol{u}_2) \cdots q_k(\boldsymbol{u}_k) \xrightarrow{\beta_k}_{\mathbb{O}} q_k(\boldsymbol{u}'_k) \xrightarrow{\alpha_k}_{\mathbb{O}} q(\boldsymbol{v})$$

pour certains $q_1, q_2, \ldots, q_k \in Q$, $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k \in \mathbb{O}$ et $\boldsymbol{u}'_1, \boldsymbol{u}'_2, \ldots, \boldsymbol{u}'_k \in \mathbb{O}$. Soit R l'ensemble de schémas linéaires défini au point (a). Nous avons donc $q_i(\boldsymbol{u}_i) \xrightarrow{\rho_i}_{\mathbb{N}^2} q_i(\boldsymbol{u}'_i)$ pour un certain $\rho_i \in R(q_i, q_i)$. Ainsi, nous observons que π peut être remplacé par un chemin provenant du schéma linéaire $\alpha_0\rho_1\alpha_1 \ldots \rho_k\alpha_k$.

Nous exploitons cette observation afin de conclure la preuve en définissant X comme étant

$$X \stackrel{\text{déf}}{=} \{\alpha_0\rho_1\alpha_1 \cdots \rho_k\alpha_k \text{ un schéma linéaire } : \\ \alpha_i \in T^*,\ |\alpha_i| \leq |Q|,\ k \leq |Q|,\ \rho_i \in R\}\ .$$

Soit $\rho \in X$, nous avons $|\rho| \leq |Q| \cdot (|Q|+1) + |Q| \cdot (|Q| + \|T\|)^{O(1)} = (|Q| + \|T\|)^{O(1)}$, et $|\rho|_* \leq 2 \cdot |Q|$. □

4.4.4 Accessibilité dans $\mathbb{N}^2$

Grâce aux résultats établis précédemment, nous prouvons maintenant le résultat principal de cette section, c'est-à-dire, que les 2-VASS sont aplatissables au sens du théorème 4.1. Pour ce faire, nous montrerons comment toute exécution restreinte à $\mathbb{N}^2$ peut être factorisée en un petit nombre d'exécutions des types (1), (2) et (3) illustrés plus tôt à la figure 4.4.

Preuve du théorème 4.1. Soient $\mathcal{V} = (Q, T)$ un 2-VASS et $D \leq (|Q| + \|T\|)^{O(1)}$ la

constante obtenue pour $\mathcal{V}$ à la proposition 4.11. Nous posons

$$\begin{aligned}
\mathbb{L} &\stackrel{\text{déf}}{=} ([0, D + \|T\|] \times \mathbb{N}) \cup (\mathbb{N} \times [0, D + \|T\|]) \ , \\
\mathbb{O} &\stackrel{\text{déf}}{=} [D, \infty)^2 \ , \text{ et} \\
\mathbb{B} &\stackrel{\text{déf}}{=} \mathbb{L} \cap \mathbb{O} = ([D, D + \|T\|] \times \mathbb{N}) \cup (\mathbb{N} \times [D, D + \|T\|]) \ .
\end{aligned}$$

L'ensemble $\mathbb{L}$ correspond aux vecteurs que nous avons qualifiés jusqu'ici de « près des axes » et $\mathbb{O}$ correspond aux vecteurs que nous avons qualifiés de « loin des axes ». Notons que l'intersection de ces régions $\mathbb{B} = \mathbb{L} \cap \mathbb{O}$ est non vide.

Faisons le sommaire de ce que nous avons prouvé sur les trois types d'exécutions :

- Par la proposition 4.6, les exécutions du type (1) sont capturées par un ensemble fini $Y_{\mathbb{L}}$ de schémas linéaires tel que pour tout $\rho \in Y_{\mathbb{L}}$, $|\rho| \leq (|Q| + \|T\| + D)^{O(1)} = (|Q| + \|T\|)^{O(1)}$ et $|\rho|_* \leq 2$.

- Par la proposition 4.11 (b), les exécutions du type (2) sont capturées par un ensemble fini X de schémas linéaires tel que pour tout $\rho \in X$, $|\rho| \leq (|Q| + \|T\|)^{O(1)}$ et $|\rho|_* \leq 2 \cdot |Q|$.

- Par la proposition 4.11 (a), les exécutions du type (3) sont capturées par un ensemble fini R de schémas linéaires tel que pour tout $\rho \in R$, $|\rho| \leq (|Q| + \|T\|)^{O(1)}$ et $|\rho|_* \leq 2$.

Soient $p(\boldsymbol{u}), q(\boldsymbol{v}) \in \mathbb{N}^2$ et $\pi \in \mathsf{Chemins}_{\mathcal{V}}(p, q)$ tels que $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{N}^2} q(\boldsymbol{v})$, où $\pi = t_1 \cdots t_k \in T^k$ et

$$p(\boldsymbol{u}) = q_0(\boldsymbol{u}_0) \xrightarrow{t_1}_{\mathbb{N}^2} q_1(\boldsymbol{u}_1) \ \cdots \ \xrightarrow{t_k}_{\mathbb{N}^2} q_k(\boldsymbol{u}_k) = q(\boldsymbol{v}) \ .$$

Nous sommes intéressés par les indices i tels que $\boldsymbol{u}_i \in \mathbb{B}$. Nous posons $I \stackrel{\text{déf}}{=} \{i \in [0, k] : \boldsymbol{u}_i \in \mathbb{B}\}$. Soit $\mathsf{suiv} : I \to I$ la fonction qui associe à chaque $i \in I$ le plus petit

élément de I plus grand que i (ou i si $i = \max(I)$), c.-à-d.

$$\mathsf{suiv}(i) \stackrel{\text{déf}}{=} \begin{cases} \min\{j \in I : j > i\} & \text{si } i < \max(I), \\ i & \text{sinon .} \end{cases}$$

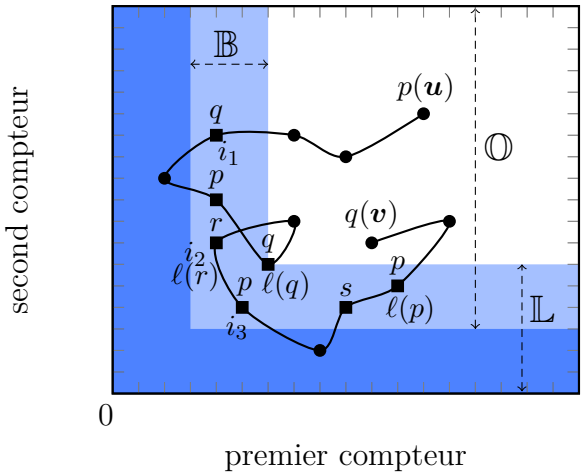


Figure 4.7 – Exemple de la décomposition d'une exécution en trois types d'exécutions tel qu'expliqué dans la preuve du théorème 4.1. Dans cet exemple, $I = \{3, 5, 6, 8, 9, 11, 12\}$ est représenté par des carrés, et $i_1 = 3$, $\ell(q) = 6$, $i_2 = \ell(r) = 8$, $i_3 = 9$ et $\ell(p) = 12$.

Nous définissons également la fonction $\ell : \{q_i \in Q : i \in I\} \to I$ qui associe à chaque q le plus grand indice i tel que $q_i = q$, c.-à-d., $\ell(q) \stackrel{\text{déf}}{=} \max\{i \in I : q_i = q\}$. Par définition de $\mathbb{O}$, $\mathbb{L}$ et $\mathbb{B}$, et par le principe du pigeonnier il existe des indices $i_1, \ldots, i_h \in I$ tels que $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{N}^2} q(\boldsymbol{v})$ peut être factorisée sous la forme (voir figure 4.7) :

$$\begin{aligned} q_0(\boldsymbol{u}_0) \xrightarrow{\pi_{0,1}}_{\mathbb{D}_0} q_{i_1}(\boldsymbol{u}_{i_1}) \xrightarrow{\pi_1}_{\mathbb{N}^2} q_{\ell(q_{i_1})}(\boldsymbol{u}_{\ell(q_{i_1})}) \xrightarrow{\pi_{1,2}}_{\mathbb{D}_1} q_{i_2}(\boldsymbol{u}_{i_2}) \xrightarrow{\pi_2}_{\mathbb{N}^2} q_{\ell(q_{i_2})}(\boldsymbol{u}_{\ell(q_{i_2})}) \\ \cdots \xrightarrow{\pi_{h-1,h}}_{\mathbb{D}_{h-1}} q_{i_h}(\boldsymbol{u}_{i_h}) \xrightarrow{\pi_h}_{\mathbb{N}^2} q_{\ell(q_{i_h})}(\boldsymbol{u}_{\ell(q_{i_h})}) \xrightarrow{\pi_{h,h+1}}_{\mathbb{D}_h} q_k(\boldsymbol{u}_k) \ , \end{aligned}$$

où

(a) $h \leq |Q|$,

(b) $i_t \in I$, et par conséquent $\boldsymbol{u}_{i_t} \in \mathbb{B}$ et $q_{i_t} = q_{\ell(q_{i_t})}$ pour tout $t \in [h]$,

(c) $\mathbb{D}_t \in \{\mathbb{O}, \mathbb{L}\}$ pour tout $t \in [0, h]$, et

(d) $i_{t+1} = \mathsf{suiv}(\ell(q_{i_t}))$ pour tout $t \in [h-1]$.

Par les points (c) et (d), chaque exécution de la forme $\xrightarrow{\pi_{t,t+1}}_{\mathbb{D}_t}$ est une exécution du type (1) ou de type (2), ainsi $\pi_{t,t+1}$ peut être remplacé par un schéma linéaire de $Y_{\mathbb{L}} \cup X$ par les propositions 4.6 et 4.11 (b). Par le point (b), chaque exécution de la forme $q_{i_t}(\boldsymbol{u}_{i_t}) \xrightarrow{\pi_t}_{\mathbb{N}^2} q_{\ell(q_{i_t})}(\boldsymbol{u}_{\ell(q_{i_t})})$ est une exécution du type (3), ainsi π_t peut être remplacé par un schéma linéaire de R puisque $\mathbb{B} \subseteq \mathbb{O}$ par la proposition 4.11 (a).

En résumé, l'exécution $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{N}^2} q(\boldsymbol{v})$ peut être remplacée par un schéma linéaire ρ, obtenu par concaténation finie de schémas linéaires de $X \cup Y_{\mathbb{L}} \cup R$, tel que $|\rho| \leq (h+1) \cdot (|Q| + \|T\|)^{O(1)} = (|Q| + \|T\|)^{O(1)}$ et $|\rho|_* \leq (h+1) \cdot 2 \cdot |Q| \leq O(|Q|^2)$. □

4.5 De l'aplatissement vers la complexité

Maintenant que nous savons aplatir les 2-VASS, nous montrons comment en dériver des bornes sur la taille des chemins témoignant de l'accessibilité. Par souci de généralité, nous montrons comment obtenir de telles bornes pour les d-VASS aplatissables pour tout $d \geq 1$. À cette fin, nous exploitons des résultats sur la taille des solutions de systèmes d'équations diophantiennes linéaires que nous introduisons maintenant.

4.5.1 Équations diophantiennes

Un système d'équations (resp. d'inégalités) diophantiennes linéaires est un système d'équations de la forme $\mathbf{A}\boldsymbol{x} = \boldsymbol{c}$ (resp. $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{c}$) où $\mathbf{A} \in \mathbb{Z}^{d \times k}$ et $\boldsymbol{c} \in \mathbb{Z}^d$ pour certains $d, k \in \mathbb{N}_{>0}$, et où les variables $\boldsymbol{x}$ doivent prendre des valeurs entières non négatives, c.-à-d. dans $\mathbb{N}^k$. Nous donnons quelques bornes utiles sur la taille des solutions de tels systèmes.

Proposition 4.15 ([Sch98, p. 239])**.** *Soit $\mathcal{I} : \mathbf{A} \cdot \boldsymbol{x} \geq \boldsymbol{c}$ un système d'inégalités diophantiennes linéaires, où $\mathbf{A} \in \mathbb{Z}^{d \times k}$. Si $\mathcal{I}$ possède une solution, alors il possède également une solution $\boldsymbol{e} \in \mathbb{N}^k$ telle que $\|\boldsymbol{e}\| \leq 2^{k^{O(1)}} \cdot O(\|\mathbf{A}\| + \|\boldsymbol{c}\|)$.*

Proposition 4.16 ([Pot91, théorème 1])**.** *Soit $\mathcal{E} : \mathbf{A} \cdot \boldsymbol{x} = \mathbf{0}$ un système d'équations diophantiennes linéaires, où $\mathbf{A} \in \mathbb{Z}^{d \times k}$. Il existe $P \subseteq \mathbb{N}^k$ tel que $\mathsf{Cone}_{\mathbb{N}}(P)$ est l'ensemble des solutions de $\mathcal{E}$, et $\|P\| \leq (\|\mathbf{A}\| + 1)^d$.*

Nous étendons la proposition ci-dessus au cas non homogène.

Corollaire 4.17. *Soit $\mathcal{E} : \mathbf{A} \cdot \boldsymbol{x} = \boldsymbol{c}$ un système d'équations diophantiennes linéaires, où $\mathbf{A} \in \mathbb{Z}^{d \times k}$. Si $\mathcal{E}$ possède une solution, alors il possède également une solution $\boldsymbol{e} \in \mathbb{N}^k$ telle que $\|\boldsymbol{e}\| \leq (\|\mathbf{A}\| + \|\boldsymbol{c}\|)^{O(d)}$.*

Démonstration. Soit $\mathcal{E}'$ le système d'équations diophantiennes linéaires suivant :

$$\mathcal{E}' \ : \ \begin{bmatrix} \mathbf{A} & -\boldsymbol{c} \end{bmatrix} \begin{pmatrix} \boldsymbol{x} \\ y \end{pmatrix} \ = \ \mathbf{0}$$

où $\boldsymbol{x} \in \mathbb{N}^k$ et $y \in \mathbb{N}$. Nous remarquons que $\boldsymbol{x}$ est une solution de $\mathcal{E}$, si, et seulement si, $(\boldsymbol{x}, 1)$ est une solution de $\mathcal{E}'$. Par la proposition 4.16, l'ensemble des solutions de $\mathcal{E}'$ est $\mathsf{Cone}_{\mathbb{N}}(P)$ pour un certain ensemble $P \subseteq \mathbb{N}^{k+1}$ tel que $\|P\| \leq (\|\mathbf{A}\| + \|\boldsymbol{c}\| + 1)^d$. Supposons que $\mathcal{E}$ possède une solution $\boldsymbol{x}$, alors $(\boldsymbol{x}, 1) \in \mathsf{Cone}_{\mathbb{N}}(P)$ et par conséquent $\exists \boldsymbol{p} \in P$ tel que $\boldsymbol{p}(k+1) = 1$. Nous notons que chaque vecteur de P est une solution de $\mathcal{E}'$, ainsi $\boldsymbol{p}$ est une solution de $\mathcal{E}$. Puisque $\|p\| \leq (\|\mathbf{A}\| + \|\boldsymbol{c}\| + 1)^d$, cela complète la preuve. □

4.5.2 Exécutions des d-VASS aplatissables

Avant de construire des systèmes d'équations diophantiennes à partir de l'aplatissement des d-VASS, nous observons d'abord que la possibilité d'exécuter un cycle plusieurs fois dans un VASS ne dépend que de la possibilité de l'exécuter à la première itération et à la derniere itération.

Proposition 4.18. *Soient $\mathcal{V} = (Q, T)$ un d-VASS et $\beta \in \mathsf{Chemins}_{\mathcal{V}}(q, q)$. Pour tout $e \in \mathbb{N}$ et tous $q(\boldsymbol{u}) \in Q \times \mathbb{N}^d$, $q(\boldsymbol{u}) \xrightarrow{\beta^{e+1}}_{\mathbb{N}^d} q(\boldsymbol{u}_{e+1})$ si, et seulement si,*

(i) $q(\boldsymbol{u}) \xrightarrow{\beta}_{\mathbb{N}^d} q(\boldsymbol{u}_1)$, *et*

(ii) $q(\boldsymbol{u}_e) \xrightarrow{\beta}_{\mathbb{N}^d} q(\boldsymbol{u}_{e+1})$

où $\boldsymbol{u}_m \stackrel{déf}{=} \boldsymbol{u} + m \cdot \delta(\beta)$.

Démonstration. Soient $e \in \mathbb{N}$ et $q(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^d$. Si $q(\boldsymbol{u}) \xrightarrow{\beta^{e+1}}_{\mathbb{N}^d} q(\boldsymbol{u}_{e+1})$ alors, par définition de $\rightarrow_{\mathbb{N}^d}$, les points (i) et (ii) sont satisfaits. Supposons donc que les points (i) et (ii) sont satisfaits et montrons que

$$q(\boldsymbol{u}) \xrightarrow{\beta^{e+1}}_{\mathbb{N}^d} q(\boldsymbol{u}_{e+1}) \ . \tag{4.2}$$

Afin d'obtenir une contradiction, supposons que (4.2) soit faux. Il existe donc un $0 \leq i \leq e$ minimal tel que $q(\boldsymbol{u}_i) \xrightarrow{\beta}_{\mathbb{N}^d} q(\boldsymbol{u}_{i+1})$ soit faux. Par les points (i) et (ii), nous avons $0 < i < e$. De plus, il existe $j \in [d]$ tel que $\boldsymbol{u}_i(j) < \boldsymbol{u}_0(j)$. Ainsi, $\delta(\beta)(j) < 0$, et par conséquent $\boldsymbol{u}_e(j) < \boldsymbol{u}_i(j)$. Or, $q(\boldsymbol{u}_e) \xrightarrow{\beta}_{\mathbb{N}^d} q(\boldsymbol{u}_{e+1})$, ce qui est une contradiction. □

Grace à la proposition précédente, nous pouvons montrer comment construire un système d'inégalités diophantiennes qui caractérise les exécutions d'un schéma linéaire.

Lemme 4.19. *Soient $\mathcal{V} = (Q, T)$ un d-VASS, $p(\boldsymbol{u}) \in Q \times \mathbb{N}^d$, et $\rho = \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$ un schéma linéaire de p vers q. Il existe un système d'inégalités diophantiennes linéaires $\mathcal{I} : \mathbf{A}\boldsymbol{x} \geq \boldsymbol{c}$ tel que*

(i) pour tout $\boldsymbol{e} \in \mathbb{Z}^d$, $\boldsymbol{e}$ est une solution de $\mathcal{I}$ si, et seulement si, $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{N}^d} q(\boldsymbol{u} + \delta(\pi))$ où $\pi = \alpha_0 \beta_1^{\boldsymbol{e}(1)+1} \alpha_1 \cdots \beta^{\boldsymbol{e}(k)+1} \alpha_k$.

(ii) $\mathbf{A} \in \mathbb{Z}^{d \cdot k \times k}$, *et*

(iii) $\|\mathbf{A}\| \leq k \cdot |\rho| \cdot \|T\|$ *et* $\|\boldsymbol{c}\| \leq O(\|\boldsymbol{u}\| + |\rho| \cdot \|T\|)$.

Démonstration. Intuitivement, nous devons construire $\mathcal{I}$ de telle sorte qu'aucun compteur ne chute sous zéro pour tout préfixe de ρ. Afin de s'assurer que α_0 puisse

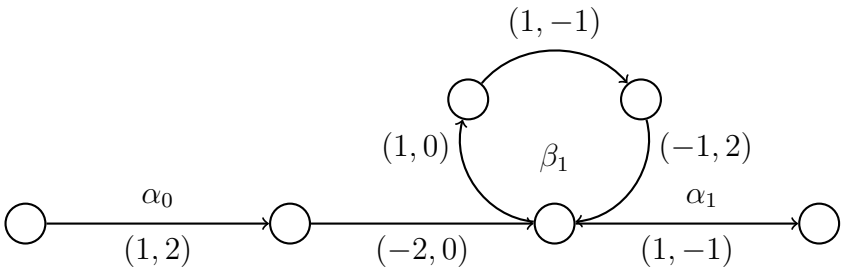


Figure 4.8 – Exemple de schéma linéaire $\alpha_0\beta_1^*\alpha_1$ duquel extraire un système d'inégalités diophantiennes linéaires.

être exécuté, il suffit de vérifier que, pour tout $1 \leq \ell \leq |\alpha_0|$,

$$\boldsymbol{u} + \delta(\alpha_0(1)\alpha_0(2)\cdots\alpha_0(\ell)) \geq \mathbf{0} \ .$$

Si $\mathsf{Cycles}(\rho)$ était vide, cela suffirait, or, ce n'est généralement pas le cas. Afin de traiter les cycles, nous pouvons exploiter la proposition 4.18. Par exemple, afin de s'assurer que β_1 puisse être exécuté $e+1$ fois, nous vérifions que, pour tout $1 \leq \ell \leq |\beta_1|$,

$$\begin{aligned} &\boldsymbol{u} + \delta(\alpha_0) + \qquad\qquad \delta(\beta_1(1)\beta_1(2)\cdots\beta_1(\ell)) \geq \mathbf{0} \ , \\ &\boldsymbol{u} + \delta(\alpha_0) + e\cdot\delta(\beta) + \delta(\beta_1(1)\beta_1(2)\cdots\beta_1(\ell)) \geq \mathbf{0} \ . \end{aligned}$$

À titre d'exemple, considérons le schéma linéaire $\rho = \alpha_0\beta_1^*\alpha_1$ illustré à la figure 4.8.

En suivant cette approche, les contraintes générées pour ρ sont :

$$\begin{aligned}
\boldsymbol{u} + \quad (1,2) &\geq \mathbf{0}\\
\boldsymbol{u} + (-1,2) &\geq \mathbf{0}\\
\boldsymbol{u} + (-1,2) + (1,\quad 0) &\geq \mathbf{0}\\
\boldsymbol{u} + (-1,2) + (2,-1) &\geq \mathbf{0}\\
\boldsymbol{u} + (-1,2) + (1,\quad 1) &\geq \mathbf{0}\\
\boldsymbol{u} + (-1,2) + e\cdot(1,1) + (1,\quad 0) &\geq \mathbf{0}\\
\boldsymbol{u} + (-1,2) + e\cdot(1,1) + (2,-1) &\geq \mathbf{0}\\
\boldsymbol{u} + (-1,2) + e\cdot(1,1) + (1,\quad 1) &\geq \mathbf{0}\\
\boldsymbol{u} + (-1,2) + (e+1)\cdot(1,1) + (1,-1) &\geq \mathbf{0}
\end{aligned}$$

Notons que plusieurs contraintes générées sont redondantes. En effet, en ramenant les termes constants à droite, il suffit de garder les valeurs maximales dans $\mathbb{Z}$ des membres de droite associés à un même membre de gauche d'une inégalité. Le système de l'exemple est ainsi équivalent à

$$\begin{aligned}
\mathbf{0} &\geq -\boldsymbol{u} + (1,-1)\\
e(1,1) &\geq -\boldsymbol{u} + (0,-1)
\end{aligned}$$

De plus, la première contrainte ne dépend d'aucune variable, elle peut ainsi être vérifiée au préalable. Si cette contrainte est satisfaite, elle peut être retirée du système, sinon il suffit de produire un système trivialement insatisfaisable, p. ex. $[\mathbf{0}]\boldsymbol{x} \geq \mathbf{1}$.

De façon générale, nous ajoutons donc à $\mathcal{I}$, les contraintes suivantes. Pour chacun des α_j où $j \in [0,k]$, nous ajoutons, pour tout $1 \leq \ell \leq |\alpha_j|$, la contrainte

$$\boldsymbol{u} + \underbrace{\sum_{0\leq i<j} \delta(\alpha_i) + (\boldsymbol{x}(i)+1)\cdot\delta(\beta_{i+1})}_{\text{correspond à } \alpha_0\beta_1^{\boldsymbol{x}(1)+1}\alpha_1 \,\cdots\, \beta_j^{\boldsymbol{x}(j)+1}} + \delta(\alpha_j(1)\alpha_j(2)\cdots\alpha_j(i)) \geq \mathbf{0}$$

Pour chacun des β_j où $j \in [k]$, nous ajoutons, pour tout $1 \leq \ell \leq |\alpha_j|$, les contraintes

$$\underbrace{\boldsymbol{u} + \sum_{0 \leq i < j} \delta(\alpha_i) + \sum_{1 \leq i < j} (\boldsymbol{x}(i)+1) \cdot \delta(\beta_{i+1})}_{\text{correspond à } \alpha_0 \beta_1^{\boldsymbol{x}(1)+1} \cdots \beta_{j-1}^{\boldsymbol{x}(j-1)+1} \alpha_{j-1}} + \qquad \delta(\beta_j(1)\beta_j(2)\cdots\beta_j(\ell)) \geq \mathbf{0}$$

$$\underbrace{\boldsymbol{u} + \sum_{1 \leq i < j} \delta(\alpha_i) + \sum_{1 \leq i < j} (\boldsymbol{x}(i)+1) \cdot \delta(\beta_{i+1}) + \boldsymbol{x}(j) \cdot \delta(\beta_j)}_{\text{correspond à } \alpha_0 \beta_1^{\boldsymbol{x}(1)+1} \cdots \beta_{j-1}^{\boldsymbol{x}(j-1)+1} \alpha_{j-1} \beta_j^{\boldsymbol{x}(j)}} + \delta(\beta_j(1)\beta_j(2)\cdots\beta_j(\ell)) \geq \mathbf{0}$$

Par construction et par la proposition 4.18, nous observons que le point (i) est satisfait. Nous devons maintenant argumenter que les points (ii) et (iii) sont également satisfaits. Remarquons d'abord que nous avons construit beaucoup plus que $d \cdot k$ contraintes. Néanmoins, il est possible d'en extraire un système $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{c}$ où

$$\mathbf{A} \overset{\text{déf}}{=} \begin{pmatrix} \delta(\beta_1) & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \delta(\beta_1) & \delta(\beta_2) & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \delta(\beta_1) & \delta(\beta_2) & \delta(\beta_3) & \dots & \delta(\beta_k) \end{pmatrix}$$

Cela est dû au fait que les coefficients de $\boldsymbol{x}(1), \boldsymbol{x}(2), \dots, \boldsymbol{x}(k)$ dans chaque contrainte correspondent à une ligne de $\mathbf{A}$. Ainsi, nous procédons comme à l'exemple mentionné plus tôt en choisissant $\boldsymbol{c}$ de telle sorte que $\boldsymbol{c}(i)$ est la plus grande valeur que le terme de droite prend dans chaque contrainte. Nous obtenons ainsi une matrice $\mathbf{A}$ de taille $d \cdot k \times k$ telle que $\|\mathbf{A}\| = k \cdot \max\{\|\delta(\beta_i)\| : i \in [k]\} \leq k \cdot |\rho| \cdot \|T\|$. De plus, les valeurs possibles des composantes de $\boldsymbol{c}$ sont les sommes partielles des déplacements de $\alpha_0\beta_1\beta_1\alpha_1 \cdots \beta_k\beta_k$ plus $\boldsymbol{u}$. Ainsi, $\|\boldsymbol{c}\| \leq \|\boldsymbol{u}\| + 2 \cdot |\rho| \cdot \|T\|$. □

Nous étendons le lemme précédent aux exécutions où le vecteur final est fixé.

Lemme 4.20. *Soient $\mathcal{V} = (Q, T)$ un d-VASS, $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^d$, et $\rho = \alpha_0\beta_1^*\alpha_1 \cdots \beta_k^*\alpha_k$ un schéma linéaire de p vers q. Il existe un système d'inégalités diophantiennes linéaires $\mathcal{I} : \mathbf{A}\boldsymbol{x} \geq \boldsymbol{c}$ tel que*

(i) *pour tout* $\boldsymbol{e} \in \mathbb{Z}^d$, $\boldsymbol{e}$ *est une solution de* $\mathcal{I}$ *si, et seulement si,* $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{N}^d} q(\boldsymbol{v})$ *où* $\pi = \alpha_0 \beta_1^{\boldsymbol{e}(1)+1} \alpha_1 \cdots \beta^{\boldsymbol{e}(k)+1} \alpha_k$.

(ii) $\mathbf{A} \in \mathbb{Z}^{d(k+2)\times k}$, *et*

(iii) $\|\mathbf{A}\| \leq k \cdot |\rho| \cdot \|T\|$ *et* $\|\boldsymbol{c}\| \leq O(\|\boldsymbol{u}\| + \|\boldsymbol{v}\| + |\rho| \cdot \|T\|)$.

Démonstration. Soient $\mathbf{B}\boldsymbol{y} \geq \boldsymbol{d}$ le système obtenu au lemme 4.19 pour $\mathcal{V}, p(\boldsymbol{u})$ et ρ. Afin de s'assurer que le vecteur atteint est bien $\boldsymbol{v}$, nous ajoutons la contrainte $\boldsymbol{u} + \delta(\alpha_0\alpha_1 \cdots \alpha_k) + \sum_{i=1}^{k} \boldsymbol{x}(i) \cdot \delta(\beta_i) = \boldsymbol{v}$. Rappelons que

$$\mathbf{B} = \begin{pmatrix} \delta(\beta_1) & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \delta(\beta_1) & \delta(\beta_2) & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \delta(\beta_1) & \delta(\beta_2) & \delta(\beta_3) & \dots & \delta(\beta_k) \end{pmatrix}$$

Nous produisons donc le système suivant,

$$\begin{pmatrix} \delta(\beta_1) & \mathbf{0} & \dots & \mathbf{0} \\ \delta(\beta_1) & \delta(\beta_2) & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \delta(\beta_1) & \delta(\beta_2) & \dots & \delta(\beta_k) \\ \delta(\beta_1) & \delta(\beta_2) & \dots & \delta(\beta_k) \\ -\delta(\beta_1) & -\delta(\beta_2) & \dots & -\delta(\beta_k) \end{pmatrix} \boldsymbol{x} \geq \begin{bmatrix} \boldsymbol{d} \\ \boldsymbol{v} - \delta(\alpha_0\alpha_1 \cdots \alpha_k) \\ -\boldsymbol{v} + \delta(\alpha_0\alpha_1 \cdots \alpha_k) \end{bmatrix}$$

Le point (i) est ainsi satisfait par construction et par le lemme 4.19. Le point (ii) est satisfait puisque $\|\mathbf{A}\| = k \cdot \max\{\|\delta(\beta_i)\| : i \in [k]\} \leq k \cdot |\rho| \cdot \|T\|$. De plus, le point (iii) est satisfait puisque $\|\boldsymbol{c}\| = \max(\|\boldsymbol{d}\|, \|\boldsymbol{v} - \delta(\alpha_0\alpha_1 \cdots \alpha_k)\|)$. □

Les deux lemmes précédents nous permettent de borner la taille des chemins minimaux dans les d-VASS aplatissables. Nous formalisons ce fait dans les deux propositions suivantes.

Proposition 4.21. *Soit $\mathcal{V} = (Q, T)$ un d-VASS et $\rho = \alpha_0\beta_1^*\alpha_1 \cdots \beta_k^*\alpha_k$ un schéma linéaire de p vers q. Pour tous $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{N}^d$, $p(\boldsymbol{u}) \xrightarrow{\rho}_{\mathbb{N}^d} q(\boldsymbol{v})$ si, et seulement si, il existe $\boldsymbol{e} \in \mathbb{N}^d$ tel que $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{N}^d} q(\boldsymbol{v})$ et $\|\boldsymbol{e}\| \leq 2^{k^{O(1)}} \cdot O(k \cdot |\rho| \cdot \|T\| + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|)$ où $\pi = \alpha_0\beta_1^{\boldsymbol{e}(1)}\alpha_1 \cdots \beta_k^{\boldsymbol{e}(k)}\alpha_k$.*

Démonstration. Soient $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{N}^d$. Supposons que $p(\boldsymbol{u}) \xrightarrow{\rho}_{\mathbb{N}^d} q(\boldsymbol{v})$. Il existe donc $\boldsymbol{e} \in \mathbb{N}^k$ tel que $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{N}^d} q(\boldsymbol{v})$ où $\pi = \alpha_0\beta_1^{\boldsymbol{e}(1)}\alpha_1 \cdots \beta_k^{\boldsymbol{e}(k)}\alpha_k$. En retirant chaque cycle β_i de ρ tel que $\boldsymbol{e}(i) = 0$, nous obtenons un nouveau schéma linéaire $\rho' = \alpha'_0\gamma_1^*\alpha'_1 \cdots \gamma_\ell^*\alpha'_\ell$ où chaque α'_i est la concaténation d'un ou plusieurs α_j, et chaque γ_i est égal à un certain β_j. Ainsi, $|\rho'| \leq |\rho|$, $\ell \leq k$, et il existe $\boldsymbol{e}' \in \mathbb{N}^\ell$ tel que $p(\boldsymbol{u}) \xrightarrow{\pi'}_{\mathbb{N}^d} q(\boldsymbol{v})$ où

$$\pi' = \alpha'_0\gamma_1^{\boldsymbol{e}'(1)+1}\alpha'_1 \cdots \gamma_\ell^{\boldsymbol{e}'(\ell)+1}\alpha'_\ell \ .$$

Soit $\mathcal{I} : \mathbf{A}\boldsymbol{x} \geq \boldsymbol{c}$ le système d'inégalités diophantiennes linéaires obtenues par le lemme 4.20 pour $\mathcal{V}$, $p(\boldsymbol{u})$, $q(\boldsymbol{v})$ et ρ'. Par le lemme 4.20, $\boldsymbol{e}'$ est une solution de $\mathcal{I}$. Ainsi, par la proposition 4.15, $\mathcal{I}$ possède une autre solution $\boldsymbol{e}'' \in \mathbb{N}^\ell$ telle que $\|\boldsymbol{e}''\| \leq 2^{\ell^{O(1)}} \cdot O(\|\mathbf{A}\| + \|\boldsymbol{c}\|)$. Nous avons donc

$$\begin{aligned}
\|\boldsymbol{e}''\| &\leq 2^{\ell^{O}(1)} \cdot O(\|\mathbf{A}\| + \|\boldsymbol{c}\|) \\
&\leq 2^{\ell^{O(1)}} \cdot O(\ell \cdot |\rho'| \cdot \|T\| + \|\boldsymbol{u}\| + \|\boldsymbol{v}\| + |\rho'| \cdot \|T\|) \\
&\leq 2^{k^{O(1)}} \cdot O(k \cdot |\rho| \cdot \|T\| + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|)
\end{aligned}$$

Par le lemme 4.20, $p(\boldsymbol{u}) \xrightarrow{\pi''}_{\mathbb{N}^d} q(\boldsymbol{v})$ où $\pi'' = \alpha'_0\gamma_1^{\boldsymbol{e}''(1)+1}\alpha'_1 \cdots \gamma_\ell^{\boldsymbol{e}''(\ell)+1}\alpha'_\ell$. Afin de conclure, notons que $\boldsymbol{e}''$ peut naturellement être étendu à $\boldsymbol{a} \in \mathbb{N}^k$ de telle sorte que

$$p(\boldsymbol{u}) \xrightarrow{\alpha_0\beta_1^{\boldsymbol{a}(1)}\alpha_1 \cdots \beta_k^{\boldsymbol{a}(k)}\alpha_k}_{\mathbb{N}^d} q(\boldsymbol{v}) \ .$$

En effet, posons $\boldsymbol{a}(i) = \boldsymbol{e}''(j) + 1$ si β_i correspond à γ_j, et $\boldsymbol{a}(i) = 0$ si β_i n'apparaît pas dans ρ'. Notons que $\|\boldsymbol{a}\| \leq \|\boldsymbol{e}''\| + 1$. □

Proposition 4.22. *Soit $\mathcal{V} = (Q, T)$ un d-VASS aplatissable grâce à un ensemble fini S de schémas linéaires. Posons $\ell_{max} \stackrel{déf}{=} \max\{|\rho| : \rho \in S\}$ et $k_{max} \stackrel{déf}{=} \max\{|\rho|_* : \rho \in S\}$. Pour tous $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^d$, $p(\boldsymbol{u}) \to^*_{\mathbb{N}^d} q(\boldsymbol{v})$ si, et seulement si, $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{N}^d} q(\boldsymbol{v})$ pour un certain $\pi \in \mathsf{Chemins}_{\mathcal{V}}(p, q)$ tel que $|\pi| \leq 2^{{k_{max}}^{O(1)}} \cdot \ell_{max} \cdot O(k_{max} \cdot \ell_{max} \cdot \|T\| + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|)$.*

Démonstration. Soient $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^d$. Supposons que $p(\boldsymbol{u}) \to^*_{\mathbb{N}^d} q(\boldsymbol{v})$, alors $p(\boldsymbol{u}) \xrightarrow{\rho}_{\mathbb{N}^d} q(\boldsymbol{v})$ pour un certain $\rho = \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k \in S$. Ainsi, il existe $\boldsymbol{e} \in \mathbb{N}^k$ tel que $\|\boldsymbol{e}\| \leq 2^{k^{O(1)}} \cdot O(k \cdot |\rho| \cdot \|T\| + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|)$ et $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{N}^d} q(\boldsymbol{v})$ où $\pi = \alpha_0 \beta_1^{\boldsymbol{e}(1)} \alpha_1 \cdots \beta_k^{\boldsymbol{e}(k)} \alpha_k$. Nous avons

$$
\begin{aligned}
|\pi| &\leq (k+1)|\rho| + \|\boldsymbol{e}\| \cdot |\rho| \\
&\leq (k_{\max} + 1)\ell_{\max} + 2^{{k_{\max}}^{O(1)}} \cdot O(k_{\max} \cdot \ell_{\max} \cdot \|T\| + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|)\ell_{\max} \\
&\leq 2^{{k_{\max}}^{O(1)}} \cdot \ell_{\max} \cdot O(k_{\max} \cdot \ell_{\max} \cdot \|T\| + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|) \ .
\end{aligned}
$$

□

4.5.3 Complexité du problème d'accessibilité pour les 2-VASS

Nous nous penchons maintenant sur la complexité du problème d'accessibilité dans les 2-VASS. En exploitant les résultats de la sous-section précédente, nous bornons la taille des chemins minimaux dans les 2-VASS.

Théorème 4.23. *Soit $\mathcal{V} = (Q, T)$ un 2-VASS. Pour tous $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^2$, $p(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v})$ si, et seulement si, $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{N}^2} q(\boldsymbol{v})$ pour un certain $\pi \in \mathsf{Chemins}_{\mathcal{V}}(p, q)$ tel que $|\pi| \leq 2^{|Q|^{O(1)}} \cdot O(\|T\|^{O(1)} + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|)$.*

Démonstration. Soit S l'ensemble fini de schémas linéaires obtenu pour $\mathcal{V}$ par le théorème 4.1. Posons $\ell_{\max} \stackrel{\text{déf}}{=} \max\{|\rho| : \rho \in S\}$ et $k_{\max} \stackrel{\text{déf}}{=} \max\{|\rho|_* : \rho \in S\}$. Soient $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^2$. Supposons que $p(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v})$. Par la proposition 4.22, $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{N}^2} q(\boldsymbol{v})$ pour un certain $\pi \in \mathsf{Chemins}_{\mathcal{V}}(p, q)$ tel que $|\pi| \leq 2^{{k_{\max}}^{O(1)}} \cdot \ell_{\max} \cdot$

$O(k_{\max} \cdot \ell_{\max} \cdot \|T\| + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|)$, $\ell_{\max} \leq (|Q| + \|T\|)^{O(1)}$ et $k_{\max} \leq O(|Q|^2)$. Ainsi

$$
\begin{aligned}
|\pi| &\leq 2^{{k_{\max}}^{O(1)}} \cdot \ell_{\max} \cdot O(k_{\max} \cdot \ell_{\max} \cdot \|T\| + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|) \\
&\leq 2^{|Q|^{O(1)}} \cdot (|Q| + \|T\|)^{O(1)} \cdot O(|Q|^2 \cdot (|Q| + \|T\|)^{O(1)} \cdot \|T\| + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|) \\
&\leq 2^{|Q|^{O(1)}} \cdot O((|Q| + \|T\|)^{O(1)} + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|) \\
&\leq 2^{|Q|^{O(1)}} \cdot O(\|T\|^{O(1)} + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|) \ . \qquad \square
\end{aligned}
$$

Nous sommes maintenant en mesure d'établir la complexité du problème d'accessibilité pour les 2-VASS.

Théorème 4.24. *Le problème d'accessibilité pour les 2-VASS, où les entiers sont encodés en binaire, est dans* PSPACE.

Démonstration. Soient $\mathcal{V} = (Q, T)$ un 2-VASS et $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^2$. Par le théorème 4.23, $p(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v})$ si, et seulement si, $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{N}^2} q(\boldsymbol{v})$ pour un certain $\pi \in \mathsf{Chemins}_{\mathcal{V}}(p, q)$ tel que $|\pi| \leq 2^{|Q|^{O(1)}} \cdot O(\|T\|^{O(1)} + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|)$. Notons que

$$
\begin{aligned}
|\pi| &\leq 2^{|Q|^{O(1)}} \cdot O(\|T\|^{O(1)} + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|) \\
&\leq 2^{|Q|^{O(1)} + O(\log\|T\| + \log\|\boldsymbol{u}\| + \log\|\boldsymbol{v}\|)} \ .
\end{aligned}
$$

Soient $0 \leq i \leq |\pi|$, et $r(\boldsymbol{w}) \in Q \times \mathbb{N}^2$ tel que $p(\boldsymbol{u}) \xrightarrow{\pi(1)\pi(2)\cdots\pi(i)} r(\boldsymbol{w})$. Nous avons $\|\boldsymbol{w}\| \leq i \cdot \|T\|$. Ainsi, $|\pi|$ et la norme de chaque vecteur intermédiaire sont exponentiels dans la taille de l'entrée. Il suffit donc de construire une machine de Turing qui débute en $p(\boldsymbol{v})$ et devine de façon non déterministe une exécution de longueur exponentielle[5] vers $q(\boldsymbol{v})$. Cela requiert de stocker l'état de contrôle et le vecteur courant, or un état de contrôle peut être encodé linéairement et un vecteur intermédiaire (de norme exponentielle) peut être encodé grâce à nombre polynomial

[5]Rappelons que notre notation informelle $O(1)$ cache une constante qui est la même pour *tous* les 2-VASS. Ainsi, la machine de Turing est bien définie. Cela sera également le cas pour les preuves subséquentes.

de bits. □

Grâce aux travaux de Fearnley & Jurdziński [FJ15], nous pouvons établir la PSPACE-complétude du problème d'accessibilité pour les 2-VASS. Fearnley & Jurdziński ont démontré dans [FJ15] que le problème d'accessibilité pour les automates à un compteur borné est PSPACE-complet. Un automate à un compteur borné est un 1-VASS $\mathcal{V} = (Q, T)$ muni d'une borne $b \in \mathbb{N}$ et tel que sa relation d'accessibilité est $\rightarrow^*_{\mathbb{B}}$ où $\mathbb{B} = [0, b]$. Nous montrons comment réduire le problème d'accessibilité pour les automates à un compteur borné au problème d'accessibilité pour les 2-VASS.

Proposition 4.25. *Le problème d'accessibilité pour les 2-VASS, où les entiers sont encodés en binaire, est* PSPACE*-difficile.*

Démonstration. Soit $\mathcal{V} = (Q, T)$ un 1-VASS muni d'une borne $b \in \mathbb{N}$. Posons $\mathbb{B} = [0, b]$. Nous construisons un 2-VASS $\mathcal{V}'$ qui simule le comportement de $\mathcal{V}$ à l'aide de deux compteurs. Le premier compteur de $\mathcal{V}'$ a pour but de vérifier que le compteur de $\mathcal{V}$ ne chute pas sous zéro, alors que le second compteur de $\mathcal{V}'$ a pour but de vérifier que le compteur de $\mathcal{V}$ ne dépasse pas b.

Soit $\mathcal{V}' \stackrel{\text{déf}}{=} (Q, T')$ le 2-VASS obtenu à partir de $\mathcal{V}$ en posant $T' \stackrel{\text{déf}}{=} \{h(t) : t \in T\}$, où $h(p, z, q) \stackrel{\text{déf}}{=} (p, (z, -z), q)$. Nous définissons l'injection $\varphi : Q \times \mathbb{B} \to Q \times \mathbb{N}^2$ telle que $\varphi(q(z)) \stackrel{\text{déf}}{=} q(z, b - z)$. Nous remarquons que pour tous $p(u), q(v) \in \mathbb{Q} \times \mathbb{B}$,

$$p(u) \rightarrow^*_{\mathbb{B}} q(v) \iff \varphi(p(u)) \rightarrow^*_{\mathbb{N}^2} \varphi(q(v)) \ . \qquad \square$$

Corollaire 4.26. *Le problème d'accessibilité pour les 2-VASS, où les entiers sont encodés en binaire, est* PSPACE*-complet.*

Nous considérons maintenant la complexité du problème lorsque les entiers sont encodés en notation unaire.

Théorème 4.27. *Le problème d'accessibilité pour les 2-VASS, où les entiers sont encodés en notation unaire, est dans* NP*.*

Démonstration. Soient $\mathcal{V} = (Q, T)$ un 2-VASS et $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^2$. Soit S l'ensemble fini de schémas linéaires obtenus pour $\mathcal{V}$ au théorème 4.1. Par la proposition 4.21, $p(\boldsymbol{u}) \rightarrow^*_{\mathbb{N}^2} q(\boldsymbol{v})$ si, et seulement si, il existe $\rho = \alpha_0\beta_1^*\alpha_1 \cdots \beta_k^*\alpha_k \in S$, et $\boldsymbol{e} \in \mathbb{N}^k$ tel que

$$\|\boldsymbol{e}\| \leq 2^{k^{O(1)}} \cdot O(k \cdot |\rho| \cdot \|T\| + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|) \tag{4.3}$$

et

$$p(\boldsymbol{u}) \xrightarrow{\alpha_0\beta_1^{\boldsymbol{e_1}}\alpha_1 \cdots \beta_k^{\boldsymbol{e}(k)}\alpha_k}_{\mathbb{N}^2} q(\boldsymbol{v}) \tag{4.4}$$

Rappelons que pour tout $\rho \in S$, nous avons $|\rho| \leq (|Q| + \|T\|)^{O(1)}$ et $k \leq O(|Q|^2)$. En particulier, $|\rho|$ est de taille polynomiale en terme de l'entrée puisque les nombres sont encodés en notation unaire. Ainsi, afin de vérifier si $p(\boldsymbol{u}) \rightarrow^*_{\mathbb{N}^2} q(\boldsymbol{v})$, nous construisons une machine de Turing qui devine de façon non déterministe un schéma linéaire de p vers q de longueur polynomiale ainsi qu'un vecteur $\boldsymbol{e}$. Notons que la représentation binaire de $\boldsymbol{e}$ est de longueur polynomiale par (4.3). La machine vérifie ensuite (4.4) en exécutant $\alpha_0\beta_1^{\boldsymbol{e_1}}\alpha_1 \cdots \beta_k^{\boldsymbol{e}(k)}$ à partir de $p(\boldsymbol{u})$. Notons qu'il serait trop long d'exécuter les cycles $\beta_i^{\boldsymbol{e_i}}$, néanmoins, il est possible de vérifier la première et dernière exécution de chaque cycle en exploitant la proposition 4.18. □

Notons que puisque le problème d'accessibilité dans les graphes dirigés est NL-complet, le problème d'accessibilité pour les 2-VASS est NL-difficile lorsque les entiers sont encodés en notation unaire. Nous précisons ce résultat en montrant que le problème est déjà NL-difficile pour les 2-VASS sous forme de schéma linéaire. Plus formellement, nous disons qu'un d-VASS est *plat* s'il ne contient pas de cycles non simples.

Théorème 4.28. *Le problème suivant est* NL*-difficile :*

Entrée: *Un 2-VASS $\mathcal{V} = (Q, T)$ plat et $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^2$ où les entiers sont encodés en notation unaire.*

Question: $p(\boldsymbol{u}) \to^*_{\mathbb{N}^2} q(\boldsymbol{v})$ *?*

Démonstration. Soit $G = (U, E)$ un graphe dirigé tel que $U = \{u_1, u_1, \ldots, u_m\}$ et $E = \{e_1, e_2, \ldots, e_n\} \subseteq U \times U$. Nous construisons un 2-VASS $\mathcal{V} = (Q, T)$ à partir de G. Nous définissons $Q \stackrel{\text{déf}}{=} \{q_{i,j}, q'_{i,j} : i \in [m], j \in [n]\}$ et

$$\begin{aligned} T \stackrel{\text{déf}}{=} & \{(q_{i,j}, \mathbf{0}, q_{i,j+1}) : i \in [m], j \in [n-1]\} \cup \\ & \{(q_{i,n}, \mathbf{0}, q_{i+1,1}) : i \in [m-1], j \in [n]\} \cup \\ & \{(q_{i,j}, (-a, a-m), q'_{i,j}) : i \in [m], j \in [n], e_j = (u_a, u_b)\} \\ & \{(q'_{i,j}, \ (b, m-b) \ , q_{i,j}) : i \in [m], j \in [n], e_j = (u_a, u_b)\} \ . \end{aligned}$$

Intuitivement, la valeur $(\ell, m - \ell)$ encode l'état u_ℓ, et le cycle de $q_{i,j}$ vers lui-même correspond à l'emprunt de l'arc $e_j = (u_a, u_b)$. Ce cycle peut seulement être emprunté sur les compteurs prenant la valeur qui encode u_a. Puisque s'il existe un chemin entre deux sommets de G, il en existe un de longueur au plus m, le 2-VASS $\mathcal{V}$ est constitué d'un motif qui se répète m fois. Plus formellement, il est possible de montrer que u_m est accessible à partir de u_1 si, et seulement si, $q_{1,1}(1, m-1) \to^*_{\mathbb{N}^2} q_{m,n}(m, 0)$. Nous illustrons ce fait à l'aide d'un exemple à la figure 4.9. Dans cet exemple, le sommet u_4 est accessible à partir de u_1 en franchissant $e_1\, e_2\, e_4$. Conséquemment nous avons l'exécution suivante dans le 2-VASS,

$$\underbrace{q_{1,1}(1,3) \to_{\mathbb{N}^2} q'_{1,1}(0,0) \to_{\mathbb{N}^2} q_{1,1}(2,2)}_{\text{emprunt de } e_1} \to_{\mathbb{N}^2} \underbrace{q_{1,2}(2,2) \to_{\mathbb{N}^2} q'_{1,2}(0,0) \to_{\mathbb{N}^2} q_{1,2}(3,1)}_{\text{emprunt de } e_2}$$

$$\to_{\mathbb{N}^2} q_{1,3}(3,1) \to_{\mathbb{N}^2} \underbrace{q_{1,4}(3,1) \to_{\mathbb{N}^2} q'_{1,4}(0,0) \to q_{1,4}(4,0)}_{\text{emprunt de } e_4} \to^*_{\mathbb{N}^2} q_{4,4}(4,0) \ . \quad \square$$

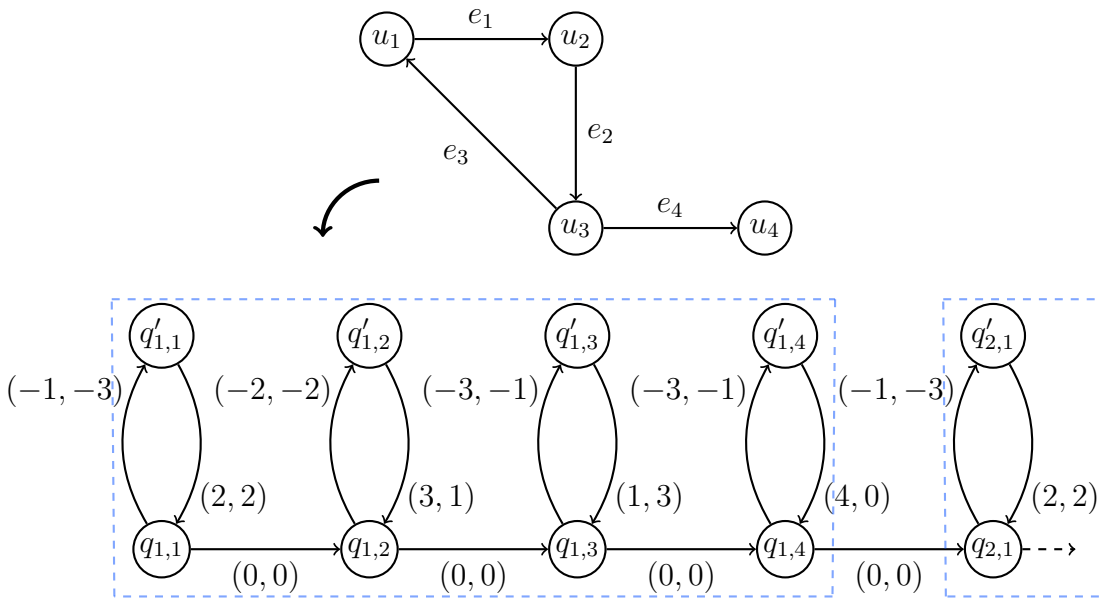


Figure 4.9 – Exemple de réduction du problème d'accessibilité dans un graphe à l'accessibilité dans un 2-VASS.

4.5.4 Complexité des problèmes de finitude et de couverture pour les d-VASS

Afin de compléter notre étude des 2-VASS, nous dressons le portrait de la complexité du problème de finitude et du problème de couverture pour les d-VASS. Nous rappelons la définition de ces problèmes introduits dans leur pleine généralité à la section 2.8.5.

FINITUDE

Entrée: Un d-VASS $\mathcal{V} = (Q, T)$ et $p(\boldsymbol{u}) \in Q \times \mathbb{N}^2$

Question: Existe-t-il $m \in \mathbb{N}$ tel que $|\mathsf{Succ}^*(p(\boldsymbol{u}))| \leq m$?

COUVERTURE

Entrée: Un d-VASS $\mathcal{V} = (Q, T)$ et $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{N}^2$.

Question: Existe-t-il $\boldsymbol{v}' \in \uparrow \boldsymbol{v}$ tel que $p(\boldsymbol{u}) \to^* q(\boldsymbol{v}')$?

La complexité de ces deux problèmes, lorsque d est fixé, a été étudiée par Rosier

& Yen in [RY86]. Ils ont démontré que le problème est PSPACE-complet pour tout $d \geq 4$. Chan [Cha88] a plus tard montré que le problème de finitude est également PSPACE-complet pour $d = 3$, laissant le cas où $d = 2$ ouvert. Il est également connu que ces problèmes sont NP-complets lorsque $d = 1$ [Haa12].

À partir de la proposition 4.25, découlant des travaux de Fearnley & Jurdziński [FJ15], il est possible de montrer que le problème de finitude et le problème de couverture sont également PSPACE-complets.

Proposition 4.29. *Le problème de finitude et le problème de couverture sont* PSPACE*-difficiles pour les 2-VASS.*

Démonstration. Nous donnons une réduction à partir du problème d'accessibilité dans les automates à un compteur borné. Soient $\mathcal{V} = (Q, T)$ un 1-VASS muni d'une borne $b \in \mathbb{N}$. Soient $p(u), q(v) \in Q \times \mathbb{B}$ où $\mathbb{B} = [0, b]$. Soit $\mathcal{V}' = (Q, T')$ le 2-VASS qui simule $\mathcal{V}$ tel que défini dans la preuve de la proposition 4.25. Nous construisons le 2-VASS

$$\mathcal{V}'' \stackrel{\text{déf}}{=} (Q \cup \{q'\}, T' \cup \{(q, (-v, v - b), q'), (q', (1, 1), q')\}) \ .$$

Nous remarquons que $p(u, b - u) \to^*_{\mathbb{N}^2} q'(0, 0)$ dans $\mathcal{V}''$ si, et seulement si, $p(u) \to^*_{\mathbb{B}} p(v)$ dans $\mathcal{V}$. Notons que la transition $(q', (1, 1), q')$ permet d'augmenter le contenu des compteurs autant que désiré. Notons également que $\mathsf{Succ}^*_{\mathcal{V}'}(p(b, b - u))$ est fini, ainsi $\mathsf{Succ}^*_{\mathcal{V}''}(p(b, b - u))$ n'est infini que si q' peut être atteint. Cela montre que

$$\begin{aligned} p(u) \to^*_{\mathbb{B}} p(v) \text{ dans } \mathcal{V} &\iff \exists \boldsymbol{v} \in \uparrow(0, 0) \text{ t.q. } p(u, b - u) \to^*_{\mathbb{N}^2} q'(\boldsymbol{v}) \text{ dans } \mathcal{V}'' \\ &\iff \mathsf{Succ}^*_{\mathcal{V}''}(p(b, b - u)) \text{ est infini.} \end{aligned}$$

□

En sommaire, nous obtenons donc :

Théorème 4.30 ([RY86, Cha88, FJ15])**.** *Les problèmes de finitude et de couverture pour les d-VASS, où les entiers sont encodés en binaire, sont*

- NP*-complets pour* $d = 1$*,*

- PSPACE-*complets pour* $d \geq 2$,

4.5.5 Complexité du problème d'accessibilité pour les d-VASS entiers

Bien que ce chapitre porte principalement sur les 2-VASS, nos résultats nous permettent de répondre à une question ouverte concernant les VASS entiers. Un *VASS entier* est simplement un VASS interprété sous sa sémantique entière, c.-à-d. dont la relation d'accessibilité considérée est $\rightarrow^*_{\mathbb{Z}^d}$. Le problème d'accessibilité pour les VASS entiers est NP-complet lorsque les entiers sont encodés en binaire, et ce même lorsque le nombre de compteur d est égal à 1 [HH14, HKOW09]. Lorsque le nombre de compteurs n'est pas fixé, c.-à-d. fait partie de l'entrée, et que les entiers sont encodés en notation unaire, le problème est également NP-complet [HH14]. La complexité du problème lorsque d est fixé et que les entiers sont encodés en notation unaire est demeurée ouverte dans [HH14]. Nous montrons que dans ce cas, le problème est NL-complet.

Théorème 4.31. *Le problème d'accessibilité pour les d-VASS entiers, où les entiers sont encodés en notation unaire, est* NL*-complet pour tout $d \geq 1$ fixé.*

Démonstration. Le problème est NL-difficile puisque le problème d'accessibilité dans les graphes dirigés s'y réduit trivialement. Soient $V = (Q, T)$ un d-VASS entier et $p(\boldsymbol{u}), q(\boldsymbol{v}) \in Q \times \mathbb{Z}^d$. Supposons que $p(\boldsymbol{u}) \xrightarrow{*}_{\mathbb{Z}^d} q(\boldsymbol{v})$. Par la proposition 4.8, il existe un schéma linéaire $\rho = \alpha_0\beta_1^*\alpha_1 \cdots \beta_k^*\alpha_k \in S$ tel que $p(\boldsymbol{u}) \xrightarrow{\rho}_{\mathbb{Z}^d} q(\boldsymbol{v})$, $|\rho| \leq 2 \cdot |Q| \cdot |T|$ et $k \leq |T|$.

Puisque nous nous intéressons à l'accessibilité sans contrainte de non négativité, nous pouvons simplement considérer le système suivant :

$$\boldsymbol{u} + \delta(\alpha_0) + \boldsymbol{x}(1) \cdot \delta(\beta_1) + \delta(\alpha_1) + \cdots + \boldsymbol{x}(k) \cdot \delta(\beta_k) + \delta(\alpha_k) = \boldsymbol{v} \ .$$

Plus formellement, soit $\mathcal{E} : A \cdot \boldsymbol{x} = \boldsymbol{c}$ le système d'équations diophantiennes linéaires

tel que $A \stackrel{\text{déf}}{=} \begin{bmatrix} \delta(\beta_1) & \cdots & \delta(\beta_k) \end{bmatrix}$ et $\boldsymbol{c} \stackrel{\text{déf}}{=} \boldsymbol{v} - (\boldsymbol{u} + \delta(\alpha_0\alpha_1\cdots\alpha_k))$. Nous avons

$$p(\boldsymbol{u}) \xrightarrow{\alpha_0\beta_1^{\boldsymbol{e}(1)}\alpha_1\cdots\beta_k^{\boldsymbol{e}(k)}\alpha_k}_{\mathbb{Z}^d} q(\boldsymbol{v})$$

si, et seulement si $\boldsymbol{e}$ est une solution de $\mathcal{E}$. Ainsi, puisque $q(\boldsymbol{v})$ est accessible par hypothèse, $\mathcal{E}$ possède une solution $\boldsymbol{e}$. Par le corollaire 4.17, $\mathcal{E}$ possède une autre solution $\boldsymbol{e}'$ telle que $\|\boldsymbol{e}'\| \leq ((|Q| + \|T\|)^{O(1)} + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|)^{O(d)}$. De plus, $p(\boldsymbol{u}) \xrightarrow{\pi}_{\mathbb{Z}^d} q(\boldsymbol{v})$ où $\pi = \alpha_0\beta_1^{\boldsymbol{e}'(1)}\alpha_1\cdots\beta_k^{\boldsymbol{e}'(k)}\alpha_k$. Ainsi,

$$\begin{aligned}
|\pi| &\leq (k+1)|\rho| + \|\boldsymbol{e}'\| \cdot |\rho| \\
&\leq (\|T\| + 1) \cdot 2 \cdot |Q| \cdot |T| + ((|Q| + \|T\|)^{O(1)} + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|)^{O(d)} \cdot 2 \cdot |Q| \cdot |T| \\
&\leq ((|Q| + \|T\|)^{O(1)} + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|)^{O(d)} .
\end{aligned}$$

De plus, pour tout $0 \leq i \leq |\pi|$

$$\begin{aligned}
\|\delta(\pi(1)\pi(2)\cdots\pi(i))\| &\leq i \cdot \|T\| \\
&\leq |\pi| \cdot \|T\| \\
&\leq ((|Q| + \|T\|)^{O(1)} + \|\boldsymbol{u}\| + \|\boldsymbol{v}\|)^{O(d)} .
\end{aligned}$$

Puisque $\|T\|, \|\boldsymbol{u}\|$ et $\|\boldsymbol{v}\|$ sont encodés en notation unaire, et que d est fixé, π et la norme de chaque vecteur intermédiaire sont de longueur polynomiale en terme de la taille de l'entrée. Ainsi, il est possible de construire une machine de Turing qui devine de façon non déterministe une exécution de taille polynomiale en stockant l'état de contrôle et le vecteur actuel. Notons qu'il est possible d'encoder les vecteurs intermédiaires avec un nombre logarithmique de bits puisque d est constant. □

4.6 Discussion

Dans ce chapitre, nous avons montré que les 2-VASS peuvent être aplatis grâce à des schémas linéaires de taille exponentielle et comportant un nombre polynomial de cycles. Cela nous a permis de montrer que le problème d'accessibilité pour les 2-VASS est PSPACE-complet lorsque les entiers sont encodés en binaire. La question de la complexité précise du problème d'accessibilité dans les 2-VASS demeurait sans réponse depuis les travaux de [RY86] en 1986. Nous avons aussi montré que le problème d'accessibilité pour les 2-VASS se situe entre NL et NP lorsque les entiers sont encodés en notation unaire. Nous avons également souligné que les problèmes de finitude et de couverture sont PSPACE-complets pour les 2-VASS, et que le problème d'accessibilité est NL-complet pour les d-VASS entiers lorsque d est fixé et que les entiers sont encodés en notation unaire.

Une question qui est demeurée ouverte est la complexité exacte du problème d'accessibilité pour les 2-VASS lorsque les entiers sont encodés en notation unaire. Cependant, en exploitant nos résultats, Lazić *et al.* [ELT16] ont montré tout récemment que ce problème est en fait NL-complet. Leroux *et al.* ont également annoncé dans [LST15] pouvoir utiliser nos résultats afin de montrer que le problème de couverture pour les systèmes d'addition de vecteurs à un compteur et une pile, mais sans états de contrôles, est dans EXPSPACE.

D'un point de vue pratique, bien que nous ayons établi la complexité précise du problème d'accessibilité pour les 2-VASS, notre approche mène à un algorithme qui doit deviner une exécution de façon non déterministe. Or, bien que facile à implémenter, cela a peu de chance d'être efficace en pratique. Ainsi, l'algorithme d'Hopcroft & Pansiot [HP79] constitue, à priori, une meilleure avenue d'implémentation, et ce malgré le fait qu'il fonctionne en temps doublement exponentiel dans le pire cas. Il serait intéressant de revisiter les résultats de ce chapitre ainsi que l'algorithme d'Hopcroft & Pansiot afin d'être en mesure de développer un outil efficace. Notons qu'une implémentation élémentaire de l'algorithme d'Hopcroft & Pansiot, développée dans le cadre de cette thèse, suggère que cet algorithme s'avère

également lent. Ainsi, des heuristiques devront être développées.

D'un point de vue théorique, il serait bien sûr intéressant d'établir la complexité du problème d'accessibilité pour les 3-VASS, ou des restrictions de 3-VASS. À ce jour, ce problème est connu comme étant PSPACE-difficile et dans F_{ω^3}. Autrement dit, les meilleurs bornes connues proviennent des 2-VASS et du cas général. Le problème s'avère cependant beaucoup plus compliqué que le cas des 2-VASS puisque les 3-VASS ne sont pas tous aplatissables [HP79, LS04] ; dans notre preuve, c'est précisément le lemme 4.12 qui ne peut être généralisé à $d = 3$ [LS04, remarque 5.2]. À court terme, une piste plus prometteuse consiste à étudier la complexité du problème d'accessibilité pour des extensions des 2-VASS. En effet, Finkel & Sutre ont démontré qu'il existe certaines extensions des 2-VASS pour lesquels le problème d'accessibilité est décidable [FS00], sans établir de bornes de complexité. Ces extensions incluent, par exemple, la classe des 2-VASS où il est possible de tester si le premier compteur vaut zéro. Finalement, la décidabilité ou la complexité du problème d'accessibilité pour les 2-VASS paramétrés pourra être étudiée. Un d-VASS paramétré est un d-VASS dont certaines transitions sont étiquetées par des variables plutôt que des vecteurs. Dans ce cas, le problème d'accessibilité de $p(\boldsymbol{u})$ à $q(\boldsymbol{v})$ consiste à vérifier s'il existe une affectation des variables qui permet d'atteindre $q(\boldsymbol{v})$ à partir de $p(\boldsymbol{u})$. Ce problème est décidable [HKOW09] pour les 1-VASS avec test à zéro, et Lechner a récemment démontré que le problème se situe dans la classe NEXPTIME [Lec15].

5

UNE APPROCHE CONTINUE AU PROBLÈME DE COUVERTURE POUR LES RÉSEAUX DE PETRI

Dans ce chapitre, nous étudions le problème de couverture pour les réseaux de Petri. Le problème de couverture est une relaxation du problème d'accessibilité étudié plus en détails au chapitre précédent dans le contexte des systèmes d'addition de vecteurs. Dans le cas du problème de couverture, plutôt que de chercher à vérifier s'il est possible d'atteindre un marquage $\boldsymbol{m}$ à partir d'un marquage initial $\boldsymbol{m}_0$, nous cherchons à vérifier s'il est possible d'atteindre un marquage plus grand ou égal à $\boldsymbol{m}$ à partir de $\boldsymbol{m}_0$. Plus formellement, le problème est défini de la façon suivante :

COUVERTURE

Entrée: Un réseau de Petri $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{N}^P$.

Question: Existe-t-il $\boldsymbol{m}' \in \uparrow \boldsymbol{m}$ tel que $\boldsymbol{m}_0 \to^* \boldsymbol{m}'$?

Ce problème a intéressé les chercheurs et les chercheuses notamment parce que plusieurs propriétés de sûreté de systèmes peuvent être vérifiées grâce à des tests de couverture. Par exemple, dans une approche mise au point par German et Sistla [GS92], des programmes concurrents, possédant un nombre arbitraire de processus à états finis qui communiquent à l'aide d'actions synchrones, sont modélisés grâce à des réseaux de Petri[1]. Chaque place du réseau de Petri correspond à une ligne du programme, et le nombre de jetons dans une place indique le nombre de fils d'exécution qui sont à la ligne associée. Il est ainsi possible de détecter, par exemple, si une propriété d'exclusion mutuelle peut être violée lorsqu'un nombre arbitraire de fils d'exécution sont exécutés en parallèle.

Dans ce chapitre, nous mettrons au point un nouvel algorithme permettant de

[1] Dans l'article, ces programmes sont modélisés de façon équivalente par des systèmes d'addition de vecteurs.

résoudre le problème de couverture pour les réseaux de Petri. Notre approche est principalement basée sur des résultats théoriques récents concernant l'accessibilité dans les réseaux de Petri continus, ainsi que les avancées pratiques de la dernière décennie dans le domaine des solveurs de safistaisabilité de formules logiques. En particulier, nous ferons usage de solveurs SMT, une généralisation des solveurs SAT qui permet de vérifier la satisfaisabilité de formules logiques dans certaines théories logiques telles que l'arithmétique de Presburger ou la théorie des nombres réels (ou rationnels). Nous montrerons qu'une implémentation de notre algorithme surpasse des outils disponibles pour le problème de couverture.

À la section 5.1, nous faisons un bref survol de la complexité du problème de couverture. Ensuite, à la section 5.2, nous décrivons un algorithme connu résolvant le problème de couverture et nous discutons de son implémentation. À la section 5.3, nous présentons une récente caractérisation de l'accessibilité dans les réseaux de Petri continus, ainsi que des algorithmes permettant de résoudre les problèmes d'accessibilité et de couverture pour les réseaux de Petri continus. À la section 5.4, nous faisons usage des concepts introduits aux sections précédentes afin de mettre au point un nouvel algorithme pour le problème de couverture. À la section 5.5, nous examinons une implémentation de notre algorithme et nous montrons comment elle se compare à certains autres outils. Nous concluons à la section 5.6 en proposant des pistes de travaux futurs.

5.1 Décidabilité et complexité du problème de couverture

Le problème de couverture fut l'un des premiers problèmes de décision pour les réseaux de Petri à être montré décidable [KM67, Hac76]. Karp & Miller donnèrent un algorithme [KM67] qui construit l'arbre d'accessibilité d'un réseau de Petri en élaguant des branches grâce à des accélérations qui permettent à l'algorithme de terminer tout en permettant d'identifier les marquages couvrables. Cette approche est décrite, par exemple, dans [GGRB09] paru en français. L'algorithme de Karp & Miller peut mener à des arbres de très grande taille, c.-à-d. de taille non

primitive récursive en fonction de la taille de l'entrée. Rackoff montra qu'il suffit en fait d'explorer les exécutions de taille doublement exponentielle [Rac78]. Ainsi, il est possible de « deviner » une exécution à la volée de façon non déterministe avec une quantité exponentielle de mémoire. Le problème de couverture est donc dans NEXPSPACE = EXPSPACE. Par conséquent, le problème de couverture est EXPSPACE-complet, puisque le problème fut démontré EXPSPACE-difficile plus tôt par Lipton *et al.* [Lip76, CLM76].

5.2 Algorithme arrière

Dans ce chapitre, nous nous pencherons sur un autre algorithme que celui de Karp & Miller : l'*algorithme arrière*. L'algorithme arrière a été utilisé pour la première fois par Arnold & Latteux [AL78] pour les VASS à un seul compteur et avec remise à zéro. Il a ensuite été introduit par Abdulla *et al.* [ACJT96], puis repris par Finkel & Schnoebelen [FPS01], dans le cadre plus général des systèmes de transitions bien structurés (voir section 3.3.4).

Algorithme 5.1 : Algorithme arrière résolvant le problème de couverture pour les réseaux de Petri.

Entrées : Réseau de Petri $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{N}^P$
Sorties : $\exists \boldsymbol{m}' \in \uparrow \boldsymbol{m}$ tel que $\boldsymbol{m}_0 \to^* \boldsymbol{m}'$?

```
1 M ← {m}
2 tant que ↑Pred(↑M) ⊈ ↑M faire
3     M ← MinBase(M ∪ ↑Pred(↑M))
4 retourner m_0 ∈ ↑M
```

L'algorithme arrière (algorithme 5.1), calcule les prédecesseurs immédiats de $\uparrow \boldsymbol{m}$, puis les prédecesseurs immédiats de $\uparrow \mathsf{Pred}(\uparrow \boldsymbol{m})$, $\uparrow \mathsf{Pred}(\uparrow \mathsf{Pred}(\uparrow \boldsymbol{m}))$ et ainsi de suite. Autrement dit, après k itérations, l'algorithme a construit l'ensemble des marquages pouvant couvrir $\boldsymbol{m}$ à l'aide d'un chemin de longueur au plus k. Plutôt que de stocker cet ensemble potentiellement infini, l'algorithme stocke sa base minimale[2] M. Lorsqu'il n'y a plus de nouveaux marquages générés, l'algorithme

[2]Il n'est pas nécessaire de stocker la base *minimale*, une base finie suffit.

se termine. Si $\boldsymbol{m}_0 \in \uparrow M$, on en conclut que $\boldsymbol{m}$ est couvrable, sinon $\boldsymbol{m}$ n'est pas couvrable.

Puisque les réseaux de Petri sont bien structurés, la terminaison de cette procédure est garantie, tel que mentionné à la section 3.3.4. Néanmoins, l'algorithme subit la EXPSPACE-complétude du problème. En effet, il a été démontré par Bozzelli & Ganty [BG11] qu'il existe des familles de réseaux de Petri pour lesquelles l'algorithme prend un nombre doublement exponentiel d'itérations, et pour lesquelles la taille de la base M obtenue est de taille doublement exponentielle.

5.2.1 Détails d'implémentation

L'algorithme arrière tel que présenté à l'algorithme 5.1 ne mène pas directement à une implémentation, p. ex. comment implémenter le calcul de $\mathsf{Pred}(\uparrow M)$? Afin de l'implémenter, nous adoptons l'approche présentée par Finkel & Leroux [FL15] (algorithme 5.2). Soient $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri, $\boldsymbol{u} \in \mathbb{N}^P$ et $t \in T$. Nous associons au marquage $\boldsymbol{u}$ et à la transition t, le marquage $\boldsymbol{u}_t$ défini par

$$\boldsymbol{u}_t(p) \stackrel{\text{déf}}{=} \max\{\mathbf{Pre}(p,t), \boldsymbol{u}(p) - \mathbf{Incid}(p,t)\} \text{ pour tout } p \in P \ .$$

Il est possible de vérifier que $\{\boldsymbol{u}_t\}$ est la base minimale des marquages couvrant $\boldsymbol{u}$ après l'activation de t, c.-à-d.

$$\{\boldsymbol{u}_t\} = \mathsf{MinBase}(\{\boldsymbol{v} \in \mathbb{N}^P : \exists \boldsymbol{v}' \in \uparrow \boldsymbol{u} \text{ t.q. } \boldsymbol{v} \xrightarrow{t} \boldsymbol{v}'\}) \ .$$

Soit $M \subseteq \mathbb{N}^P$, nous posons $\mathsf{PreBase}(M) \stackrel{\text{déf}}{=} \bigcup_{\boldsymbol{u} \in M, t \in T} \{\boldsymbol{u}_t\}$ ce qui nous permet d'obtenir la base finie $\mathsf{PreBase}(M)$, potentiellement non minimale, de $\uparrow \mathsf{Pred}(\uparrow M)$, c.-à-d.

$$\begin{aligned} \uparrow \mathsf{PreBase}(M) &= \{\boldsymbol{v} \in \mathbb{N}^P : \exists \boldsymbol{v}' \in \uparrow M \text{ t.q. } \boldsymbol{v} \to \boldsymbol{v}'\} \\ &= \uparrow \mathsf{Pred}(\uparrow M) \ . \end{aligned}$$

Cela nous permet donc d'implémenter le calcul de $\uparrow \mathsf{Pred}(\uparrow M)$ aux lignes 2 et 3 de l'algorithme 5.1.

Afin d'implémenter l'inclusion et la minimisation de bases d'ensembles clos par le haut, respectivement aux lignes 2 et 3 de l'algorithme 5.1, nous référons le lecteur à la section 2.4.2. Nous pouvons également améliorer la boucle principale, à la ligne 2 de l'algorithme 5.1, en vérifiant à chaque itération si $\boldsymbol{m}_0 \in \uparrow M$ plutôt que d'attendre à la toute fin ; cela n'aura aucune incidence si $\boldsymbol{m}$ n'est pas couvrable, mais pourra accélérer considérablement le temps d'exécution si $\boldsymbol{m}$ est couvrable.

Algorithme 5.2 : Implémentation alternative de l'algorithme 5.1.

Entrées : Réseau de Petri $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{N}^P$
Sorties : $\exists \boldsymbol{m}' \in \uparrow \boldsymbol{m}$ tel que $\boldsymbol{m}_0 \to^* \boldsymbol{m}'$?

1 $M \leftarrow \{\boldsymbol{m}\}$
2 **tant que** $\boldsymbol{m}_0 \notin \uparrow M$ **faire**
3 $B \leftarrow \mathsf{PreBase}(M) \setminus \uparrow M$
4 **si** $B = \emptyset$ **alors**
5 **retourner** *faux*
6 **sinon**
7 $M \leftarrow \mathsf{MinBase}(M \cup B)$
8 **retourner** *vrai*

Plutôt que de calculer une base minimale à partir des bases M et $\mathsf{PreBase}(M)$, comme à la ligne 3 de l'algorithme 5.1, nous proposons de calculer, de façon équivalente, d'abord $B \stackrel{\text{déf}}{=} \mathsf{MinBase}(\mathsf{PreBase}(M) \setminus \uparrow M)$, puis $\mathsf{MinBase}(M \cup B)$. Cela permet de mettre en évidence les nouveaux marquages générés, c.-à-d. $\uparrow B$, ce qui nous permettra plus tard d'implémenter des heuristiques afin d'éliminer de nouveaux marquages jugés inutiles. De plus, cela permet souvent d'accélérer considérablement le temps de calcul de la minimisation de base qui est coûteuse en pratique.

5.3 Approximer l'accessibilité à l'aide des réseaux de Petri continus

Le problème d'accessibilité pour les réseaux de Petri consiste à déterminer si $\boldsymbol{m}_0 \to^* \boldsymbol{m}$, sur entrée $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{N}^P$. Puisque les problèmes

d'accessibilité pour les réseaux de Petri et les systèmes d'addition de vecteurs sont équivalents sous réduction log-espace (section 2.9.3), la complexité du problème d'accessibilité se trouve donc entre EXPSPACE et F_{ω^3} (chapitre 4). Le problème d'accessibilité est beaucoup plus simple à résoudre lorsque la sémantique des réseaux de Petri est modifiée afin de permettre des marquages et des activations de transitions fractionnaires. Dans cette section, nous étudions une telle variante des réseaux de Petri, nommée réseaux de Petri continus [DA87], qui permet d'approximer la relation d'accessibilité. Cela nous permettra d'adapter l'algorithme arrière.

5.3.1 Réseaux de Petri continus

Les réseaux de Petri continus, introduits par David & Alla [DA87, DA10], permettent notamment d'approximer les systèmes discrets au profit de faciliter leur analyse. Par exemple, [CSMS10] décrit une méthode basée sur les réseaux de Petri continus qui permet de diagnostiquer les erreurs dans les systèmes manufacturiers plus efficacement que dans le cas discret. Un réseau de Petri continu est un réseau de Petri dont la sémantique permet les marquages rationnels[3], ainsi que l'activation fractionnaire des transitions. Plus formellement,

Definition 5.1. Un *marquage* d'un réseau de Petri continu $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ est un vecteur $\boldsymbol{m} \in \mathbb{Q}_{\geq 0}^{P}$. Soient $t \in T$ et $\boldsymbol{m} \in \mathbb{Q}_{\geq 0}^{P}$, nous définissons $\mathsf{DegActivation}(t, \boldsymbol{m}) \in \mathbb{Q}_{\geq 0} \cup \{\infty\}$ comme étant :

$$\mathsf{DegActivation}(t, \boldsymbol{m}) \stackrel{\text{déf}}{=} \begin{cases} \min \left\{ \dfrac{\boldsymbol{m}(p)}{\mathbf{Pre}(p,t)} : p \in {}^\bullet t \right\} & \text{si } {}^\bullet t \neq \emptyset \ , \\ \infty & \text{sinon.} \end{cases}$$

Nous disons que $t \in T$ est $\mathbb{Q}$-*franchissable* en $\boldsymbol{m}$ lorsque $\mathsf{DegActivation}(t, \boldsymbol{m}) > 0$. Une telle transition $\mathbb{Q}$-franchissable peut être *activée* d'une quantité $\alpha \in \mathbb{Q}_{\geq 0}$ telle que $0 < \alpha \leq \mathsf{DegActivation}(t, \boldsymbol{m})$, auquel cas elle mène au nouveau marquage $\boldsymbol{m}'$

[3] La définition originale permet les marquages réels, mais nous nous limiterons aux marquages rationnels dans cette thèse. Ces deux définitions sont montrées équivalentes dans [FH15].

tel que, pour tout $p \in P$,

$$\begin{aligned} \boldsymbol{m}'(p) &= \boldsymbol{m} - \alpha \cdot \mathbf{Pre}(p,t) + \alpha \cdot \mathbf{Post}(p,t) \\ &= \boldsymbol{m} + \alpha \cdot \mathbf{Incid}(p,t) \ . \end{aligned}$$

Nous associons à $\mathcal{N}$ le système de transitions ordonné $\mathcal{S} = (\mathbb{Q}^d_{\geq 0}, \to, \mathbb{Q}_{>0} \times T, \leq)$ tel que $\boldsymbol{m} \xrightarrow{\alpha\, t} \boldsymbol{m}'$ lorsque t est $\mathbb{Q}$-franchissable en $\boldsymbol{m}$, et $\boldsymbol{m}'$ est le marquage obtenu en activant t de la quantité $0 < \alpha \leq \mathsf{DegActivation}(t, \boldsymbol{m})$ en $\boldsymbol{m}$. Notons que les réseaux de Petri continus ne sont pas bien structurés puisque $\leq$ n'est pas un beau préordre pour $\mathbb{Q}^d_{\geq 0}$. Ils possèdent néanmoins la monotonicité forte et stricte des systèmes de transitions bien structurés. À la figure 5.1, nous illustrons un réseau de Petri continu $\mathcal{N}$ initialement marqué par $(1, 0)$. En activant son unique transition t d'une quantité $\frac{1}{2}$, nous obtenons le marquage $(\frac{1}{2}, \frac{1}{2})$. Notons qu'en activant t à nouveau d'une quantité $\frac{1}{4}$, puis $\frac{1}{8}$ et ainsi de suite, nous nous rapprochons de plus en plus du marquage $(0, 1)$ sans jamais l'atteindre. Ces comportements limites sont décrits dans [FH15], mais ne seront pas considérés dans notre étude.

5.3.2 Accessibilité dans les réseaux de Petri continus

Les réseaux de Petri continus permettent d'approximer l'accessibilité des réseaux de Petri au sens suivant :

Proposition 5.2. *Si $\boldsymbol{m}_0 \to^* \boldsymbol{m}$ dans un réseau de Petri (standard), alors $\boldsymbol{m}_0 \to^* \boldsymbol{m}$ sous la sémantique continue.*

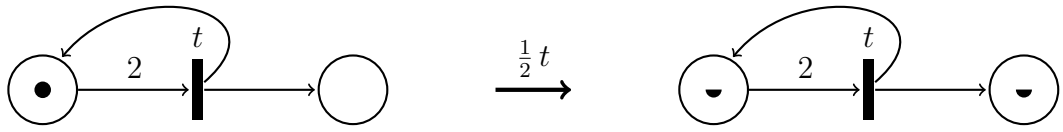


Figure 5.1 – Exemple de réseau de Petri continu $\mathcal{N}$. *Gauche :* $\mathcal{N}$ est initialement marqué par $(1, 0)$; *droite :* après l'activation de la transition t par une quantité $\frac{1}{2}$, le marquage $(\frac{1}{2}, \frac{1}{2})$ est obtenu.

Ici, le terme *sémantique continue* d'un réseau de Petri $\mathcal{N}$ signifie simplement d'interpéter $\mathcal{N}$ comme étant un réseau de Petri continu. Nous utiliserons parfois également le terme *sémantique standard* afin de signifier que nous interprétons un réseau de Petri continu de façon discrète.

Notons que la réciproque de la proposition 5.2 est généralement fausse. Cette proposition nous apprend donc que lorsqu'un marquage est non accessible sous la sémantique continue, il ne l'est pas non plus sous la sémantique standard. Nous utiliserons ce fait afin d'améliorer l'algorithme arrière.

Il a été démontré par Recalde *et al.* [RTS99] que le problème d'accessibilité pour les réseaux de Petri continus est soluble en temps polynomial lorsque les réseaux respectent certaines propriétés. Júlvez *et al.* [JRS03] montrèrent que le problème est dans EXPTIME dans le cas général. Fraca & Haddad [FH15] montrèrent que le problème est en fait P-complet.

L'algorithme de Fraca & Haddad découle de la caractérisation simple suivante de la relation $\to^*$, basée sur les ensembles dits d'activation :

Definition 5.3. Soient $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri (standard ou continu) et $\boldsymbol{m}_0$ un marquage. Nous définissons l'*ensemble d'activation* de $\boldsymbol{m}_0$ comme étant l'ensemble $\mathsf{Activ}(\boldsymbol{m}_0) \subseteq 2^T$ défini par

$$\mathsf{Activ}(\boldsymbol{m}_0) \stackrel{\text{déf}}{=} \{[\![\mathsf{Parikh}(\pi)]\!] : \pi \in \mathsf{Chemins}(\boldsymbol{m}_0)\} \ .$$

En d'autres termes, $T \in \mathsf{Activ}(\boldsymbol{m}_0)$ si, et seulement si, il existe un chemin π et un marquage $\boldsymbol{m}$ tels que $\boldsymbol{m}_0 \xrightarrow{\pi} \boldsymbol{m}$ et tels que π contienne précisément les transitions de T. Notons que $\emptyset \in \mathsf{Activ}(\boldsymbol{m}_0)$ puisque la séquence vide est un chemin à partir de tout marquage $\boldsymbol{m}_0$.

La caractérisation de Fraca & Haddad se formule ainsi :

Théorème 5.4 ([FH15, théorème 20])**.** *Soient $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri continu et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{N}^P$. Nous avons $\boldsymbol{m}_0 \to^* \boldsymbol{m}$ si, et seulement si, il existe $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^T$ tel que*

(i) $\boldsymbol{m} = \boldsymbol{m}_0 + \mathbf{Incid} \cdot \boldsymbol{v}$,

(ii) $[\![\boldsymbol{v}]\!] \in \mathit{Activ}_{\mathcal{N}}(\boldsymbol{m}_0)$, *et*

(iii) $[\![\boldsymbol{v}]\!] \in \mathit{Activ}_{\mathcal{N}^{-1}}(\boldsymbol{m})$.

Lorsque $\boldsymbol{m} \to^* \boldsymbol{m}_0$, il existe un chemin π tel que $\boldsymbol{m} \xrightarrow{\pi} \boldsymbol{m}_0$, et ainsi $\boldsymbol{v} \stackrel{\text{déf}}{=} \mathsf{Parikh}(\pi)$ satisfait clairement ces conditions, et ce même dans un réseau de Petri standard. Ce qui est plus étonnant ici est que ces conditions soient également suffisantes, ce qui n'est pas le cas dans les réseaux de Petri standards. Nous n'entrerons pas dans les détails de la preuve du théorème 5.4. Ici, nous nous intéressons plutôt aux façons d'exploiter ce théorème afin de décider l'accessibilité.

5.3.3 Une approche non déterministe au problème d'accessibilité

Le théorème 5.4 peut être exploité afin d'obtenir un algorithme fonctionnant en temps polynomial. Toutefois, avant de voir comment cela est possible, nous expliquons comment l'exploiter afin d'obtenir un algorithme fonctionnant en temps polynomial *non déterministe*. Bien que cela apparaisse moins efficace, nous verrons comment exploiter l'algorithme non déterministe à la section 5.4.1.

Voyons d'abord comment vérifier les conditions (ii) et (iii) du théorème 5.4. Soient $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri continu, $\boldsymbol{m}_0 \in \mathbb{N}^P$ et $T' \subseteq T$. Nous cherchons à déterminer si $T' \in \mathsf{Activ}(\boldsymbol{m}_0)$, autrement dit, s'il existe un chemin π à partir de $\boldsymbol{m}_0$ tel que $[\![\mathsf{Parikh}(\pi)]\!] = T'$. Nous nous restreignons donc à $\mathcal{N}_{T'}$ et nous cherchons à déterminer si toutes les transitions peuvent être activées dans un certain ordre.

Si l'on espère activer toutes les transitions, alors au moins une transition de T' doit être activable en $\boldsymbol{m}_0$, c.-à-d. il existe $t \in T'$ telle que ${}^\bullet t \subseteq [\![\boldsymbol{m}_0]\!]$. Remarquons que la quantité de jetons n'importe pas, puisque t pourra être activée d'une quantité α arbitrairement aussi petite que désiré. Par exemple, considérons le réseau de Petri $\mathcal{N}$ illustré à la figure 5.2. Les transitions t_2 et t_3 ne sont pas activables en $\boldsymbol{m}_0$, mais la transition t_1 l'est. En activant t_1 d'une quantité $\alpha = 1/2$, les places p_1 et p_2 ne sont

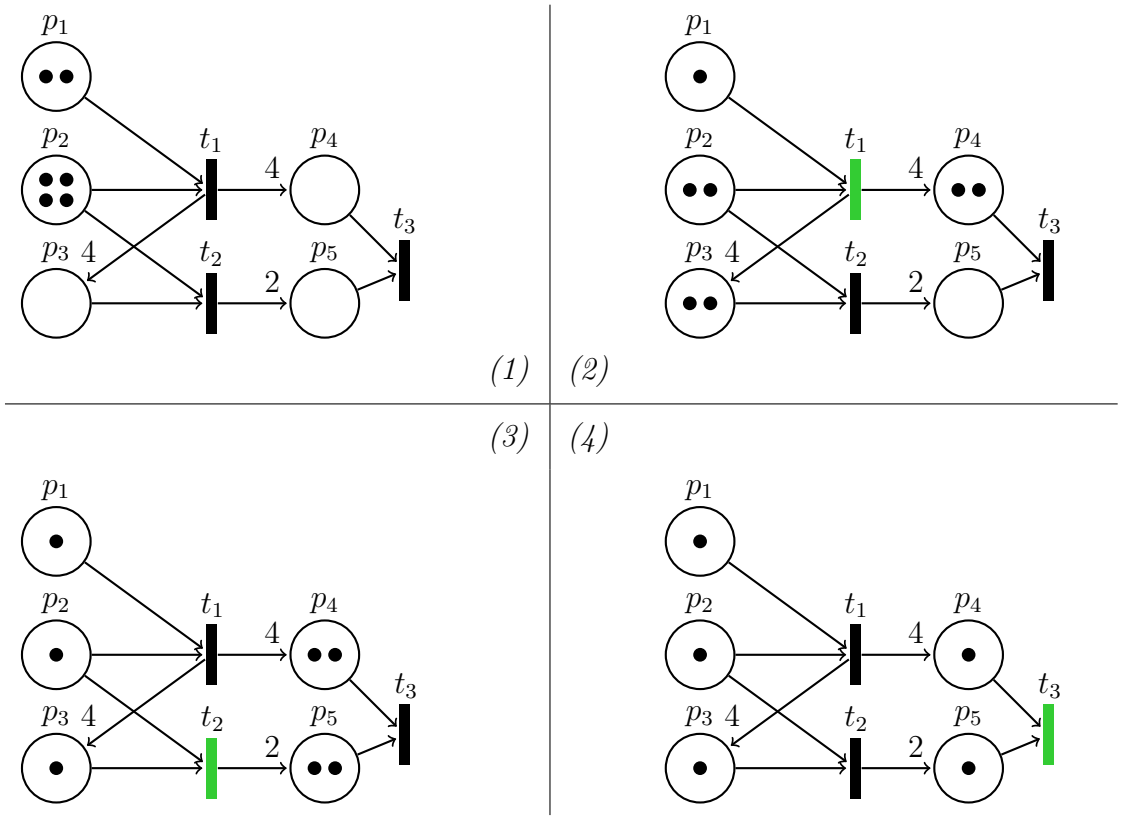


Figure 5.2 – Exemple de « propagation » du marquage $\boldsymbol{m}_0 = (2, 4, 0, 0, 0)$ dans un réseau de Petri $\mathcal{N}$ continu afin de montrer que $\{t_1, t_2, t_3\} \in \mathsf{Activ}(\boldsymbol{m}_0)$. *(1) Coin supérieur gauche :* marquage initial de $\mathcal{N}$; *(2) coin supérieur droit :* marquage obtenu après l'activation de $\frac{1}{2}\, t_1$; *(3) coin inférieur gauche :* marquage obtenu après l'activation de $\frac{1}{2}\, t_2$; *(4) coin inférieur droit :* marquage obtenu après l'activation de $\frac{1}{2}\, t_3$.

pas entièrement vidées. Ainsi, il est maintenant possible d'activer t_2 entièrement, c.-à-d. d'une quantité $\alpha = 1$. Il est ensuite possible d'activer t_3 entièrement. Cela montre donc que $T' \in \mathsf{Activ}(\boldsymbol{m}_0)$.

L'exemple décrit mène directement à une procédure ; il suffit de marquer les places dont le contenu est non nul et d'activer les transitions pouvant être activées jusqu'à ce qu'il ne soit plus possible de progresser. Cela fonctionne puisqu'en activant une transition, il est toujours possible de laisser une quantité, potentiellement fractionnaire, de jetons dans les places d'entrée.

Cette procédure est illustrée à l'algorithme 5.3 et est tirée de [FH15]. Cet

Algorithme 5.3 : Algorithme de calcul d'ensemble maximal d'activation.

Entrées : Réseau de Petri continu $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$, $\boldsymbol{m}_0 \in \mathbb{N}^P$ et $T' \subseteq T$

Sorties : Plus grand ensemble $T'' \subseteq T'$ tel que $T'' \in \mathsf{Activ}(\boldsymbol{m}_0)$

```
1  T'' ← ∅
2  P' ← ⟦m_0⟧
3  continuer ← vrai
4  tant que continuer faire
5      continuer ← faux
6      pour t ∈ T' \ T'' faire
7          si •t ⊆ P' alors
8              T'' ← T'' ∪ {t}
9              P' ← P' ∪ t•
10             continuer ← vrai
11 retourner T''
```

algorithme prend en entrée $T' \subseteq T$ et retourne T' si $T' \in \mathsf{Activ}(\boldsymbol{m}_0)$, autrement le sous-ensemble maximal $T'' \subseteq T$ tel que $T'' \in \mathsf{Activ}(\boldsymbol{m}_0)$ est retourné. Notons que ce sous-ensemble maximal est bien défini car $\mathsf{Activ}(\boldsymbol{m}_0)$ est fermé sous l'union ; cela est dû au fait que la possibilité d'activer des chemins π et π' indépendamment implique qu'il est possible d'activer $\pi'' = \frac{1}{2}\pi\ \frac{1}{2}\pi'$ (voir [FH15, lemme 17] pour une preuve formelle). Nous définissons, au passage, cet ensemble qui nous sera utile plus tard :

Definition 5.5. Soit $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri continu, et $\boldsymbol{m}_0 \in \mathbb{N}^P$. L'*ensemble d'activation maximal* est $\mathsf{ActivMax}(\boldsymbol{m}_0) \stackrel{\text{déf}}{=} \bigcup_{T' \in \mathsf{Activ}(\boldsymbol{m}_0)} T'$.

Nous illustrons le fonctionnement de l'algorithme 5.3 sur un autre exemple. Nous insistons sur le fonctionnement de cet algorithme puisqu'il sera exprimé dans une logique du premier ordre à la section 5.4.1. Soit $\mathcal{N}$ le réseau de Petri continu illustré à la gauche de la figure 5.3. Au départ, seule la place p_1 est marquée, ce qui permet d'activer la transition t_1 et de marquer p_3. Il est ensuite possible d'activer la transition t_2 et de marquer p_4. À ce stade, t_3 ne peut pas être activée, en effet, p_2 ne pourra jamais recevoir de jetons. L'algorithme 5.3 conclut donc

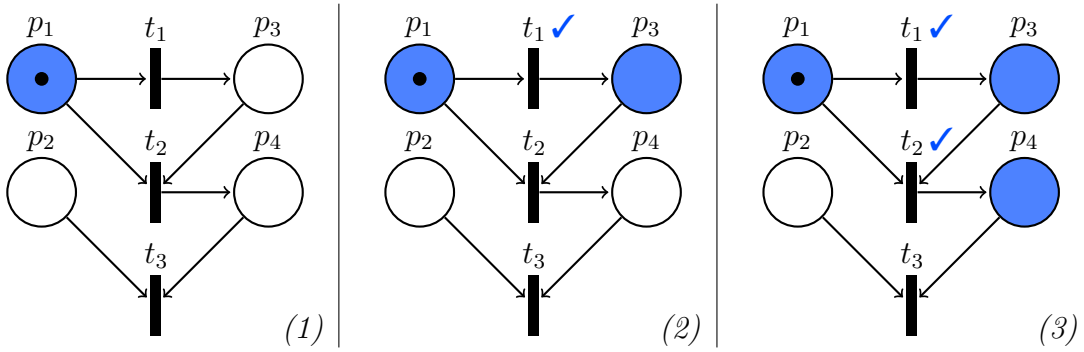


Figure 5.3 – Exécution de l'algorithme 5.3 sur un exemple de réseau de Petri continu $\mathcal{N}$. Les places dont l'arrière-plan est rempli en couleur (▬) correspondent à l'ensemble P' de l'algorithme. Les transitions identifiées par le symbole ✓correspondent à l'ensemble T'' de l'algorithme. *(1) Gauche :* la place p_1 est marquée car elle contient un jeton ; *(2) centre :* la transition t_1 peut être activée, ce qui permet de marquer la place p_3 ; *(3) droite :* la transition t_2 peut être activée, ce qui permet de marquer la place p_4. L'algorithme termine car aucune autre transition ne peut être activée.

que $\{t_1, t_2, t_3\} \notin \mathsf{Activ}(\boldsymbol{m}_0)$ et retourne plutôt $\mathsf{ActivMax}(\boldsymbol{m}_0) = \{t_1, t_2\}$. En effet, $\{t_1, t_2\} \in \mathsf{Activ}(\boldsymbol{m}_0)$ puisque $\pi = \frac{1}{2}\, t_1\ \frac{1}{4}\, t_2$ peut être activé à partir de $\boldsymbol{m}_0$.

Proposition 5.6 ([FH15])**.** *Le problème d'accessibilité pour les réseaux de Petri continus est dans* NP.

Démonstration. Soient $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri continu et $\boldsymbol{m}_0$, $\boldsymbol{m} \in \mathbb{N}^P$. Par le théorème 5.4, $\boldsymbol{m}_0 \to^* \boldsymbol{m}$ si, et seulement si, il existe $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^T$ tel que $\mathbf{Incid} \cdot \boldsymbol{v} = \boldsymbol{m} - \boldsymbol{m}_0$ et $[\![\boldsymbol{v}]\!] \in \mathsf{Activ}_{\mathcal{N}}(\boldsymbol{m}_0) \cap \mathsf{Activ}_{\mathcal{N}^{-1}}(\boldsymbol{m})$. Afin de vérifier si un tel vecteur $\boldsymbol{v}$ existe, un algorithme non déterministe choisit d'abord un support $T' \subseteq T$ et vérifie ensuite si $T' \in \mathsf{Activ}_{\mathcal{N}}(\boldsymbol{m}_0) \cap \mathsf{Activ}_{\mathcal{N}^{-1}}(\boldsymbol{m})$ à l'aide de l'algorithme 5.3. Si cette propriété est satisfaite, il reste à vérifier s'il existe $\boldsymbol{v}' \in \mathbb{Q}_{\geq 0}^{T'}$ tel que $\mathbf{Incid}[P \times T'] \cdot \boldsymbol{v}' = \boldsymbol{m} - \boldsymbol{m}_0$. Puisqu'il est possible de résoudre un système de programmation linéaire en temps polynomial [PS98, théorème 8.5], il est en particulier possible de tester si un tel vecteur $\boldsymbol{v}'$ existe en temps polynomial. □

5.3.4 De l'approche non déterministe à l'algorithme déterministe de Fraca & Haddad

Nous présentons maintenant l'algorithme polynomial de Fraca & Haddad [FH15] résolvant le problème d'accessibilité pour les réseaux de Petri continus. Une légère variante[4] de l'algorithme original est illustré à l'algorithme 5.4.

Algorithme 5.4 : Algorithme de Fraca et Haddad résolvant le problème d'accessibilité pour les réseaux de Petri continus.

Entrées : Réseau de Petri continu $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{N}^P$
Sorties : $\boldsymbol{m}_0 \nrightarrow^* \boldsymbol{m}'$?

1 **si** $\boldsymbol{m} = \boldsymbol{m}_0$ **alors retourner** *vrai*
2 $T' \leftarrow T$
3 **tant que** $T' \neq \emptyset$ **faire**
4 $\quad S \leftarrow \emptyset$
5 $\quad$ **pour** $t \in T'$ **faire**
6 $\quad\quad$ **si** $\exists \boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T'}$ t.q. $(\mathbf{Incid}[P \times T'] \cdot \boldsymbol{v} = \boldsymbol{m} - \boldsymbol{m}_0 \wedge \boldsymbol{v}(t) > 0)$ **alors**
7 $\quad\quad\quad S \leftarrow S \cup [\![\boldsymbol{v}]\!]$
8 $\quad$ **si** $S = \emptyset$ **alors retourner** *faux*
9 $\quad \boldsymbol{m}_0' \leftarrow \boldsymbol{m}_0[{}^\bullet S^\bullet]$
10 $\quad \boldsymbol{m}' \leftarrow \boldsymbol{m}[{}^\bullet S^\bullet]$
11 $\quad T' \leftarrow \mathsf{ActivMax}_{\mathcal{N}_S}(\boldsymbol{m}_0') \cap \mathsf{ActivMax}_{\mathcal{N}_S^{-1}}(\boldsymbol{m}')$
12 $\quad$ **si** $T' = S$ **alors retourner** *vrai*
13 **retourner** *faux*

Afin de tester les conditions du théorème 5.4 en temps polynomial pour un réseau de Petri continu $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$, l'algorithme 5.4 calcule implicitement un vecteur $\boldsymbol{v}_{\max} \in \mathbb{Q}_{\geq 0}^{T'}$ de support maximal satisfaisant toutes les conditions. De façon plus détaillée, l'algorithme cherche implicitement, aux lignes 4 à 8, un vecteur $\boldsymbol{v}_{\max} \in \mathbb{Q}_{\geq 0}^{T'}$ satisfaisant le point (i) du théorème 5.4 et tel que $[\![\boldsymbol{v}_{\max}]\!]$ soit maximal sous l'inclusion. Par la suite, aux lignes 9 à 11, l'algorithme vérifie si $\boldsymbol{v}_{\max}$ satisfait les points (ii) et (iii) du théorème 5.4. Pour ce faire, l'algorithme calcule

$$T' \overset{\text{déf}}{=} \mathsf{ActivMax}_{\mathcal{N}_S}(\boldsymbol{m}_0[{}^\bullet S^\bullet]) \cap \mathsf{ActivMax}_{\mathcal{N}_S^{-1}}(\boldsymbol{m}[{}^\bullet S^\bullet]) \ ,$$

[4] L'algorithme original paru dans [FH15] calcule l'image de Parikh d'un chemin de $\boldsymbol{m}_0$ vers $\boldsymbol{m}$ lorsqu'il en existe un. Ici, nous ne retournons que vrai ou faux.

où $S = [\![\boldsymbol{v}_{\max}]\!]$, à l'aide de l'algorithme 5.3. Si $[\![\boldsymbol{v}_{\max}]\!] = T'$, alors $[\![\boldsymbol{v}_{\max}]\!]$ satisfait les points (ii) et (iii), et ainsi l'algorithme retourne vrai. Autrement, l'algorithme répète le processus avec $T' \subset S$, jusqu'à ce qu'une solution soit trouvée ou que toutes les transitions soient épuisées.

Afin d'utiliser l'algorithme 5.4 de Fraca & Haddad pour résoudre le problème de couverture, il suffit de convertir $\mathcal{N}$ en $\mathcal{N}'$, le réseau de Petri continu qui peut décrémenter individuellement chaque place d'un jeton.

5.4 Algorithme arrière modulo accessibilité continue

Nous combinons maintenant les idées des sections 5.2 et 5.3 afin d'élaborer un nouvel algorithme résolvant efficacement le problème de couverture pour les réseaux de Petri. Nous étendons l'algorithme arrière qui est correct et complet en lui ajoutant des heuristiques efficaces basées sur la couverture dans les réseaux de Petri continus. Ainsi, afin de déterminer si $\boldsymbol{m}$ est couvrable à partir de $\boldsymbol{m}_0$, nous utiliserons d'abord l'algorithme 5.4 comme filtre afin de déterminer en temps polynomial si $\boldsymbol{m}$ est couvrable sous la sémantique continue. S'il ne l'est pas, il n'est pas couvrable sous la sémantique standard et l'algorithme termine. Autrement, nous poursuivons à l'aide de l'implémentation de l'algorithme arrière illustrée à l'algorithme 5.2 avec des instructions additionnelles qui s'assurent d'éliminer les nouveaux marquages B qui ne sont pas couvrables sous la sémantique continue. Pour des raisons pratiques, et afin d'exploiter le caractère itératif de l'algorithme arrière, ce filtrage de B sera réalisé grâce à l'algorithme polynomial *non déterministe* (section 5.3.3). Plus précisément, nous traduirons l'algorithme décrit à la proposition 5.6 en logique du premier ordre.

5.4.1 Exprimer l'accessibilité continue en logique du premier ordre

Avant de présenter notre algorithme, nous montrons comment décrire la relation d'accessibilité d'un réseau de Petri continu dans un fragment existentiel de la logique du premier ordre (section 2.6). Plus précisément, en se basant sur la

proposition 5.6, nous construirons des formules de $\mathsf{FO}(\mathbb{Q}_{\geq 0}, +, <)$ qui décrivent les conditions du théorème 5.4.

Fixons $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri. Dans l'ensemble des formules, $\boldsymbol{x}_{\boldsymbol{m}_0}$ et $\boldsymbol{x}_{\boldsymbol{m}}$ dénotent des vecteurs de variables du premier ordre indicées par P, et $\boldsymbol{x}_{\boldsymbol{v}}$ dénote un vecteur de variables du premier ordre indicées par T. Les vecteurs de variables $\boldsymbol{x}_{\boldsymbol{m}_0}$ et $\boldsymbol{x}_{\boldsymbol{m}}$ représenterons d'éventuels marquages $\boldsymbol{m}_0$ et $\boldsymbol{m}$ dont nous cherchons à savoir si $\boldsymbol{m}_0 \rightarrow^* \boldsymbol{m}$. Le vecteur de variables $\boldsymbol{x}_{\boldsymbol{v}}$ décrira l'image de Parikh d'un éventuel chemin $\pi \in T^*$.

Le point (i) du théorème 5.4 est décrit par la formule suivante :

$$\begin{aligned} \Phi^{(\mathrm{i})}_{\mathcal{N}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) &\stackrel{\text{déf}}{=} \boldsymbol{x} = \boldsymbol{x}_0 + \mathbf{Incid} \cdot \boldsymbol{v} \\ &= \textstyle\bigwedge_{p \in P} \left(\boldsymbol{x}(p) = \boldsymbol{x}_0(p) + \sum_{t \in T} \mathbf{Incid}(p,t) \cdot \boldsymbol{y}(t)\right) \ . \end{aligned}$$

Notons que l'expression $\mathbf{Incid}(p,t) \cdot \boldsymbol{y}(t)$ ne nécessite pas la multiplication puisque $\mathcal{N}$ est fixé et qu'ainsi $\mathbf{Incid}(p,t)$ est une constante. Il suffit donc d'écrire

$$\underbrace{\boldsymbol{y}(t) + \boldsymbol{y}(t) + \cdots + \boldsymbol{y}(t)}_{\mathbf{Incid}(p,t) \text{ fois}} \ .$$

Nous expliquons maintenant comment traduire les points (ii) et (iii) du théorème 5.4. Nous construisons une formule qui simule le fonctionnement de l'algorithme 5.3 détaillé à la section 5.3.3. La construction de cette formule est inspirée d'une technique de Verma *et al.* [VSS05] utilisée pour démontrer que la relation d'accessibilité des réseaux de Petri « communication-free » peut être définie par une formule existentielle de Presburger de taille linéaire. Plus précisément, nous introduisons un vecteur $\boldsymbol{z}$ de variables du premier ordre indicées par $P \cup T$. Ces variables représenteront l'ordre dans lequel les places et les transitions sont marquées par l'algorithme 5.3. Ainsi, $\boldsymbol{z}(t)$ (resp. $\boldsymbol{z}(p)$) indique l'itération à laquelle t (resp. p) est marquée, avec la convention que la valeur 0 signifie que t (resp. p) n'est jamais marquée.

Notons d'abord que si une transition t est utilisée, alors t doit être marquée, et

ce, après les places d'entrée de t. Cette observation s'écrit de la façon suivante :

$$\Phi_{\mathcal{N}}^{\text{trans}}(\boldsymbol{x_v}, \boldsymbol{y}) \stackrel{\text{déf}}{=} \bigwedge_{t \in T} \left(\boldsymbol{x_v}(t) > 0 \rightarrow \bigwedge_{p \in {}^\bullet t} 0 < \boldsymbol{y}(p) \leq \boldsymbol{y}(t) \right).$$

De plus, si une place p est marquée, alors ou bien elle contient initialement des jetons, ou bien elle est marquée grâce à l'activation d'une transition t, auquel cas p est marquée après t :

$$\Phi_{\mathcal{N}}^{\text{places}}(\boldsymbol{x_{m_0}}, \boldsymbol{x_v}, \boldsymbol{y}) \stackrel{\text{déf}}{=} \bigwedge_{p \in P} \left(\boldsymbol{y}(p) > 0 \rightarrow \left(\boldsymbol{x_{m_0}}(p) > 0 \vee \bigvee_{t \in {}^\bullet p} 0 < \boldsymbol{y}(t) < \boldsymbol{y}(p) \right) \right) .$$

La conjonction des deux formules précédentes décrit les points (ii) et (iii) du théorème 5.4 :

$$\Phi_{\mathcal{N}}^{(\text{ii})}(\boldsymbol{x_{m_0}}, \boldsymbol{x_v}) \stackrel{\text{déf}}{=} \exists \boldsymbol{y} : \Phi_{\mathcal{N}}^{\text{trans}}(\boldsymbol{x_v}, \boldsymbol{y}) \ \wedge \Phi_{\mathcal{N}}^{\text{places}}(\boldsymbol{x_{m_0}}, \boldsymbol{x_v}, \boldsymbol{y}) ,$$

$$\Phi_{\mathcal{N}}^{(\text{iii})}(\boldsymbol{x_m}, \boldsymbol{x_v}) \stackrel{\text{déf}}{=} \exists \boldsymbol{y} : \Phi_{\mathcal{N}^{-1}}^{\text{trans}}(\boldsymbol{x_v}, \boldsymbol{y}) \ \wedge \Phi_{\mathcal{N}^{-1}}^{\text{places}}(\boldsymbol{x_m}, \boldsymbol{x_v}, \boldsymbol{y}) .$$

Notons que ces formules ne forcent pas les indices représentés par $\boldsymbol{y}$ à être consécutifs. Or, cela n'est pas nécessaire, puisqu'un ordonnancement $i_1 < i_2 < \ldots < i_k$ valide induit naturellement un ordonnancement valide sur $[k]$.

Ainsi, en prenant la conjonction de $\Phi_{\mathcal{N}}^{(\text{i})}$, $\Phi_{\mathcal{N}}^{(\text{ii})}$ et $\Phi_{\mathcal{N}}^{(\text{iii})}$, nous obtenons une formule de $\mathsf{FO}(\mathbb{Q}_{\geq 0}, +, <)$ qui définit l'accessibilité dans $\mathcal{N}$:

$$\Phi_{\mathcal{N}}(\boldsymbol{x_{m_0}}, \boldsymbol{x_m}) \stackrel{\text{déf}}{=} \exists \boldsymbol{x_v} : \Phi_{\mathcal{N}}^{(\text{i})}(\boldsymbol{x_{m_0}}, \boldsymbol{x_m}, \boldsymbol{x_v}) \wedge \Phi_{\mathcal{N}}^{(\text{ii})}(\boldsymbol{x_{m_0}}, \boldsymbol{x_v}) \wedge \Phi_{\mathcal{N}}^{(\text{iii})}(\boldsymbol{x_m}, \boldsymbol{x_v}).$$

Formellement, nous avons montré le résultat suivant :

Proposition 5.7. *Soit $\mathcal{N}$ un réseau de Petri continu. Il existe une formule existentielle $\Phi_{\mathcal{N}}(\boldsymbol{x_{m_0}}, \boldsymbol{x_m})$ de $\mathsf{FO}(\mathbb{Q}_{\geq 0}, +, <)$, de taille linéaire, calculable en temps polynomial à partir de $\mathcal{N}$, et telle que $\boldsymbol{m_0} \rightarrow^* \boldsymbol{m} \iff \Phi_{\mathcal{N}}(\boldsymbol{m_0}, \boldsymbol{m})$.*

Le problème qui consiste à vérifier, sur entrée $\boldsymbol{m_0}, \boldsymbol{m}$, la validité de $\Phi_{\mathcal{N}}(\boldsymbol{m_0}, \boldsymbol{m})$

se trouve dans NP (p. ex., voir [Son85]). Nous verrons à la section 5.5 la raison pour laquelle nous préférons construire une telle formule plutôt que d'utiliser directement l'algorithme polynomial 5.4. Notons déjà que la validation de $\Phi_{\mathcal{N}}$ interprétée dans $\mathbb{N}$ est aussi dans NP et que cela permet de mieux approximer l'accessibilité dans les réseaux de Petri (standards). En effet, si $\Phi_{\mathcal{N}}(\boldsymbol{m}_0, \boldsymbol{m})$ n'est pas satisfaisable dans $\mathbb{N}$, alors $\boldsymbol{m}$ n'est pas accessible à partir de $\boldsymbol{m}_0$ sous la sémantique standard. Plus formellement,

Proposition 5.8. *Soient $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{N}^P$. Si $\boldsymbol{m}_0 \to^* \boldsymbol{m}$, alors il existe $\boldsymbol{v} \in \mathbb{N}^T$ tel que*

(i) $\boldsymbol{m} = \boldsymbol{m}_0 + \mathbf{Incid} \cdot \boldsymbol{v}$,

(ii) $[\![\boldsymbol{v}]\!] \in \mathit{Activ}_{\mathcal{N}}(\boldsymbol{m}_0)$, *et*

(iii) $[\![\boldsymbol{v}]\!] \in \mathit{Activ}_{\mathcal{N}^{-1}}(\boldsymbol{m})$.

Corollaire 5.9. *Soient $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{N}^P$. Nous avons $\neg\Phi_{\mathcal{N}}(\boldsymbol{m}_0, \boldsymbol{m}) \implies \neg(\boldsymbol{m}_0 \to^* \boldsymbol{m})$.*

Remarque 5.10. La proposition 5.7 permet au passage d'améliorer la meilleure borne connue pour le problème d'inclusion de réseaux de Petri continus, qui est à ce jour dans EXPTIME [FH15]. Étant donné deux réseaux de Petri continus $\mathcal{N}, \mathcal{N}'$ possédant le même ensemble de places, et $\boldsymbol{m}_0, \boldsymbol{m}'_0$ deux marquages, ce problème consiste à déterminer si l'ensemble des marquages accessibles à partir de $\boldsymbol{m}_0$ dans $\mathcal{N}$ est inclus dans l'ensemble des marquages accessibles à partir de $\boldsymbol{m}'_0$ dans $\mathcal{N}'$.

Plus formellement, nous cherchons à déterminer si $\mathsf{Succ}^*_{\mathcal{N}}(\boldsymbol{m}_0) \subseteq \mathsf{Succ}^*_{\mathcal{N}}(\boldsymbol{m})$, et ainsi si la formule $\forall \boldsymbol{m}\ \Phi_{\mathcal{N}}(\boldsymbol{m}_0, \boldsymbol{m}) \to \Phi_{\mathcal{N}'}(\boldsymbol{m}'_0, \boldsymbol{m})$ est valide. Cette formule fait partie du fragment Π_2 de $\mathrm{FO}(\mathbb{Q}_{\geq 0}, +, <)$, c'est-à-dire qu'elle possède une seule alternation de quantificateur de la forme $\forall\ \exists$. Le problème qui consiste à vérifier si une telle formule possède une solution se trouve dans la classe $\Pi^{\mathsf{P}}_2 \subseteq$ PSPACE de la hiérarchie polynomiale [Son85]. Ainsi, le problème d'inclusion de réseaux de Petri continus se trouve dans $\Pi^{\mathsf{P}}_2 \subseteq$ PSPACE.

5.4.2 Présentation de l'algorithme

Nous présentons maintenant notre nouvel algorithme résolvant le problème de couverture dans les réseaux de Petri. Nous le nommons *algorithme arrière modulo accessibilité continue* et l'illustrons à l'algorithme 5.5 où nous soulignons en couleur (et) les instructions ajoutées à l'algorithme arrière 5.2.

Algorithme 5.5 : Algorithme arrière modulo accessibilité continue.

Entrées : Réseau de Petri $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{N}^P$
Sorties : $\exists \boldsymbol{m}' \in \uparrow \boldsymbol{m}$ tel que $\boldsymbol{m}_0 \rightarrow^* \boldsymbol{m}'$?

1 $M \leftarrow \{\boldsymbol{m}\}$
2 **si** $\boldsymbol{m}$ n'est pas couvrable à partir de $\boldsymbol{m}_0$ sous la sémantique continue **alors**
3 **retourner** *faux*
4 $\psi(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y} : \Phi_{\mathcal{N}}(\boldsymbol{m}_0, \boldsymbol{y}) \wedge \boldsymbol{y} \geq \boldsymbol{x}$
5 **tant que** $\boldsymbol{m}_0 \notin \uparrow M$ **faire**
6 $B \leftarrow \mathsf{PreBase}(M) \setminus \uparrow M$
7 $D \leftarrow \{\boldsymbol{v} \in B : \neg\psi(\boldsymbol{v})\}$
8 $B \leftarrow B \setminus D$
9 **si** $B = \emptyset$ **alors**
10 **retourner** *faux*
11 **sinon**
12 $M \leftarrow \mathsf{MinBase}(M \cup B)$
13 $\psi(\boldsymbol{x}) \leftarrow \psi(\boldsymbol{x}) \wedge \bigwedge_{\boldsymbol{v} \in D} \boldsymbol{x} \not\geq \boldsymbol{v}$
14 **retourner** *vrai*

Les lignes 2 et 3 de l'algorithme 5.5 forment une heuristique exploitant la proposition 5.2 afin de détecter si $\boldsymbol{m}$ n'est pas couvrable. Bien que $\boldsymbol{m}$ puisse être non couvrable sans que cela soit détecté aux lignes 2 et 3, ce test peut être effectué en temps polynomial grâce à l'algorithe 5.4 et l'observation discutée à la fin de la sous-section 5.3.4.

La ligne 4 définit une formule ψ, à partir de $\Phi_{\mathcal{N}}$, telle que $\psi(\boldsymbol{x})$ est satisfaisable si, et seulement si, $\boldsymbol{x}$ est couvrable à partir de $\boldsymbol{m}_0$. Aux lignes 6 et 7, nous éliminons les nouveaux marquages de B qui ne sont pas couvrables à partir de $\boldsymbol{m}_0$ et qui n'auraient donc jamais mené à $\boldsymbol{m}_0$ dans les calculs subséquents. Finalement, à la ligne 13, nous mettons à jour la formule ψ en explicitant le fait que pour tout marquage $\boldsymbol{v}$ éliminé, aucun marquage de $\uparrow \boldsymbol{v}$ ne peut couvrir $\boldsymbol{m}_0$. Bien que cela

ne modifie pas l'ensemble des solutions de ψ, un telle redondance permet souvent, en pratique, d'améliorer le temps d'exécution des tests de satisfaisabilité d'une formule.

Proposition 5.11. *Soient $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{N}^P$. L'algorithme 5.5 termine toujours, et retourne vrai si, et seulement si, $\boldsymbol{m}$ est couvrable à partir de $\boldsymbol{m}_0$.*

Démonstration. Soient B_n et M_n respectivement la valeur de B et M aux lignes 8 et 12 de l'algorithme 5.5 à la $n^{\text{ème}}$ itération de la boucle `tant que`. Nous observons que

$$\uparrow M_n = \Big\{ \boldsymbol{x} \in \mathbb{N}^P : \boldsymbol{x} \text{ est couvrable à partir de } \boldsymbol{m}_0 \text{ sous la sémantique continue} \wedge \exists \boldsymbol{y} \in \uparrow \boldsymbol{m},\ \exists k \in [0, n] \text{ t.q. } \boldsymbol{x} \to^k \boldsymbol{y} \Big\}$$

$$B_n \subseteq \uparrow M_n \setminus \uparrow M_{n-1}.$$

Notons que $\boldsymbol{m}$ est couvrable à partir de $\boldsymbol{m}_0$ si, et seulement si, $\boldsymbol{m}$ est couvrable à partir de $\boldsymbol{m}_0$ sous la sémantique continue, et $\exists \boldsymbol{y} \in \uparrow \boldsymbol{m}$ tel que $\boldsymbol{m}_0 \to^* \boldsymbol{y}$. Ainsi, par définition des lignes 10 et 14, l'algorithme est correct.

Puisque $\leq$ est un beau préordre pour $\mathbb{N}^p$, nous savons que la suite $\uparrow M_1 \subseteq \uparrow M_2 \subseteq \ldots$ converge après un certain nombre n d'itérations (section 2.4.2). Ainsi, après n itérations, ou bien $B = \emptyset$ à la ligne 9, ou bien $\boldsymbol{m}_0 \in \uparrow M$ à la ligne 5. Dans les deux cas, l'algorithme termine. Ainsi, l'algorithme termine toujours. □

Remarque 5.12. Dans notre implémentation de l'algorithme 5.5, nous avons légèrement modifié l'algorithme en replaçant l'instruction $M \leftarrow \mathsf{MinBase}(M \cup B)$ de la ligne 12 par $M \leftarrow \mathsf{MinBase}(M \cup \mathsf{min}_{c,k} B)$. Ici, $c, k \in \mathbb{N}$ sont des paramètres de l'algorithme, et $\mathsf{min}_{c,k} B$ est l'ensemble des $c + |B| \div k$ éléments de B avec les plus petites « norme 1 », c.-à-d. la norme définie par $\|\boldsymbol{v}\|_1 \stackrel{\text{déf}}{=} \sum_{p \in P} |\boldsymbol{v}|$. De cette façon, des paramètres c et k choisis empiriquement permettent de créer un goulot d'étranglement qui donne priorité aux plus petits marquages qui permettent d'éliminer un plus grand nombre de marquages dans leur clôture par le haut.

Cette variation de l'algorithme est également correcte et termine toujours. Il peut être montré que l'utilisation de $\mathsf{min}_{c,k}B$, plutôt que B à la ligne 12, calcule ultimement le même ensemble $\uparrow M$ au coût possible d'augmenter le nombre d'itérations.

Nous justifions maintenant l'utilisation de $\Phi_{\mathcal{N}}$, dont le problème de satisfaisabilité est dans NP, plutôt que l'utilisation de l'algorithme polynomial de Fraca & Haddad. Aux lignes 2 et 3 de l'algorithme 5.5, nous faisons appel *une fois* à l'algorithme 5.4 de Fraca & Haddad. Par la suite, nous utilisons plutôt $\Phi_{\mathcal{N}}$ sur *plusieurs* marquages à *chaque* itération. La raison derrière cette stratégie est motivée par le fait que bien qu'un algorithme polynomial soit théoriquement préférable, une grande quantité d'appels ne s'avère pas nécessairement rapide en pratique. Tel que nous le verrons dans nos résultats expérimentaux à la section 5.5, l'algorithme de Fraca & Haddad s'avère efficace en pratique, mais il n'est pas adapté à la résolution d'un grand nombre d'instances. En revanche, de multiples requêtes de satisfaisabilité à $\Phi_{\mathcal{N}}$ profitent des stratégies internes des solveurs modernes. À chaque appel, ces solveurs raffinent leur représentation de l'espace des solutions de façon sophistiquée. C'est notamment pour cette raison que nous mettons la formule à jour à la ligne 13 de l'algorithme 5.5 ; afin de guider les solveurs dans leur recherche. De plus, tel que noté plus tôt, il nous est possible d'interpréter $\Phi_{\mathcal{N}}$ sur $\mathbb{N}$ plutôt que sur $\mathbb{Q}_{\geq 0}$ afin d'améliorer l'approximation et d'augmenter le nombre de marquages éliminés.

5.4.3 Similarités et différences avec l'approche de Esparza *et al.*

Dans [ELM+14], Esparza *et al.* ont présenté une procédure pour le problème de couverture basée sur [EM00] qui exploite l'équation $\boldsymbol{m} = \boldsymbol{m}_0 + \mathbf{Incid} \cdot \boldsymbol{v}$ ainsi qu'un type d'ensembles de places appelés *pièges*. L'approche d'Esparza *et al.* est basée sur l'approche « CEGAR », de l'anglais « Counter-Example Guided Abstraction Refinement », qui consiste à abstraire un système puis d'itérativement raffiner cette abstraction à l'aide d'information acquise en explorant un espace de solutions.

Nous détaillons l'approche d'Esparza *et al.* puisqu'elle est similaire à celle adoptée dans ce chapitre, puis nous la comparons à la nôtre. Soit $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri. Nous disons qu'un sous-ensemble non vide $Q \subseteq P$ est un *piège* de $\mathcal{N}$ si $Q^\bullet \subseteq {}^\bullet Q$, et qu'il est un *siphon* de $\mathcal{N}$ si ${}^\bullet Q \subseteq Q^\bullet$. Nous disons qu'un piège (resp. un siphon) Q est *marqué* par un marquage $\boldsymbol{m}$ lorsque $\sum_{p \in Q} \boldsymbol{m}(p) > 0$. Une propriété intéressante des pièges est que lorsqu'un piège est marqué, il demeure marqué après toute activation subséquente de transitions. Similairement, lorsqu'un siphon n'est pas marqué, il demeure non marqué après toute activation de transitions. De plus, un piège de $\mathcal{N}$ est un siphon de $\mathcal{N}^{-1}$ et vice-versa. Les conditions utilisés par Esparza *et al.* dans [ELM+14], inspirées de [EM00], peuvent être résumées de la façon suivante :

Proposition 5.13 ([ELM+14])**.** *Soient $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri standard (resp. continu) et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{N}^P$ (resp. $\in \mathbb{Q}^P_{\geq 0}$). Si $\boldsymbol{m}_0 \to^* \boldsymbol{m}$, alors il existe $\boldsymbol{v} \in \mathbb{N}^T$ (resp. $\in \mathbb{Q}^T_{\geq 0}$) tel que*

(i) $\boldsymbol{m} = \boldsymbol{m}_0 + \mathbf{Incid} \cdot \boldsymbol{v}$*, et*

(ii) pour tout piège $Q \subseteq P$, si Q est marqué par $\boldsymbol{m}_0$, alors Q est marqué par $\boldsymbol{m}$.

Comme dans notre approche, Esparza *et al.* vérifient leurs conditions à l'aide d'un solveur SMT. Le quantificateur universel de la condition (ii) est géré par une énumération incrémentale des pièges. Il est démontré dans [FH15, proposition 18] que la condition (iii) du théorème 5.4 est équivalente à ce que $\mathcal{N}^{-1}_{[\![\boldsymbol{v}]\!]}$ ne possède aucun siphon marqué par $\boldsymbol{m}$. Cette condition apparaît donc similaire à la condition (ii) de la proposition 5.13. En fait, nous pouvons montrer que les conditions du théorème 5.4 sont au moins aussi fortes que les conditions de la proposition 5.13 :

Proposition 5.14. *Soient $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ un réseau de Petri continu et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{Q}^P_{\geq 0}$. Si $\mathcal{N}, \boldsymbol{m}_0, \boldsymbol{m}$ satisfont les conditions (i) et (iii) du théorème 5.4, alors ils satisfont également les conditions (i) et (ii) de la proposition 5.13.*

Démonstration. Nous démontrons la contraposée : si pour tout $\boldsymbol{v} \in \mathbb{Q}^T_{\geq 0}$, l'une des

conditions de la proposition 5.13 n'est pas satisfaite, alors pour tout $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$, l'une de (i) ou de (iii) du théorème 5.4 n'est pas satisfaite.

Soit $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$. Supposons que $\boldsymbol{v}$ ne satisfasse pas la condition (i) de la proposition 5.13. Puisque celle-ci est identique à la condition (i) du théorème 5.4, nous avons terminé. Supposons donc que $\boldsymbol{v}$ satisfasse la condition (i) de la proposition 5.13,

$$\boldsymbol{m} = \boldsymbol{m}_0 + \mathbf{Incid} \cdot \boldsymbol{v} \ , \tag{5.1}$$

mais pas sa condition (ii). Ainsi, il existe un piège $Q \subseteq P$ dans $\mathcal{N}$ qui marque $\boldsymbol{m}_0$, mais qui ne marque pas $\boldsymbol{m}$. L'ensemble Q est donc un siphon dans $\mathcal{N}^{-1}$ qui marque $\boldsymbol{m}_0$, mais qui ne marque pas $\boldsymbol{m}$. Posons $T' = [\![\boldsymbol{v}]\!]$, $P' = {}^{\bullet}T'^{\bullet}$ et $Q' \stackrel{\text{déf}}{=} Q \cap P'$. Nous prétendons que

$$Q' \text{ est un siphon dans } \mathcal{N}_{T'}^{-1} \text{ qui n'est pas marqué par } \boldsymbol{m} \ . \tag{5.2}$$

Par [FH15, proposition 18], cela implique que $T' \notin \mathsf{Activ}(\mathcal{N}_{T'}^{-1}, \boldsymbol{m})$. Par conséquent, $[\![\boldsymbol{v}]\!] \notin \mathsf{Activ}(\mathcal{N}^{-1}, \boldsymbol{m})$, et la condition (iii) du théorème 5.4 n'est donc pas satisfaite.

Montrons (5.2). Rappelons d'abord que Q n'est pas marqué par $\boldsymbol{m}$, ainsi $Q' \subseteq Q$ n'est pas marqué par $\boldsymbol{m}$. Soit $t \in T'$ tel que $t \in {}^{\bullet}Q'$. Puisque $Q' \subseteq Q$, nous avons également $t \in {}^{\bullet}Q$. Puisque Q est un siphon dans $\mathcal{N}^{-1}$, nous avons $t \in Q^{\bullet}$. Par définition, $t \in P'^{\bullet}$, ainsi $t \in Q^{\bullet} \cap P'^{\bullet}$ et par conséquent $t \in Q'^{\bullet}$. Nous avons donc ${}^{\bullet}Q' \subseteq Q'^{\bullet}$. Afin de montrer que Q' est bien un siphon, il demeure à montrer que $Q' \neq \emptyset$. Puisque Q est marqué par $\boldsymbol{m}_0$, mais pas par $\boldsymbol{m}$, il existe $p \in Q$ tel que

$\boldsymbol{m}_0(p) > 0$ et $\boldsymbol{m}(p) = 0$. Par (5.1), $\boldsymbol{m} = \boldsymbol{m}_0 + \mathbf{Incid} \cdot \boldsymbol{v}$. Ainsi,

$$\begin{aligned}
0 &= \boldsymbol{m}_0(p) + (\mathbf{Incid} \cdot \boldsymbol{v})(p) \\
&= \boldsymbol{m}_0(p) + \sum_{t \in T} \mathbf{Incid}(p,t) \cdot \boldsymbol{v}(t) \\
&= \boldsymbol{m}_0(p) + \sum_{t \in [\![\boldsymbol{v}]\!]} \mathbf{Incid}(p,t) \cdot \boldsymbol{v}(t) && (\text{car } \boldsymbol{v}(t) = 0 \text{ pour } t \notin [\![\boldsymbol{v}]\!]) \\
&= \boldsymbol{m}_0(p) + \sum_{t \in T'} \mathbf{Incid}(p,t) \cdot \boldsymbol{v}(t) && (\text{car } T' = [\![\boldsymbol{v}]\!])
\end{aligned}$$

Puisque $\boldsymbol{m}(p) > 0$, il existe donc forcément $t \in T'$ tel que $\mathbf{Incid}(p,t) < 0$. Ainsi, $p \in {}^\bullet t^\bullet$ et par conséquent $p \in P'$. Par conséquent, $p \in Q \cap P' = Q'$, ce qui conclut la preuve. □

En fait, le théorème 5.4 est plus fort que la proposition 5.13, comme en témoigne la proposition suivante :

Proposition 5.15. *Il existe un réseau de Petri continu $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$, et $\boldsymbol{m}_0, \boldsymbol{m} \in \mathbb{Q}_{\geq 0}^P$ qui satisfont les conditions de la proposition 5.13, mais qui ne satisfont pas les conditions (i) et (iii) du théorème 5.4.*

Démonstration. Soient $\mathcal{N} = (\{p,q\}, \{s,t\}, \mathbf{Pre}, \mathbf{Post})$ le réseau de Petri continu illustré à la figure 5.4, $\boldsymbol{m}_0 = (1,0)$ et $\boldsymbol{m} = (0,1)$. Nous remarquons que $\boldsymbol{m}$ n'est

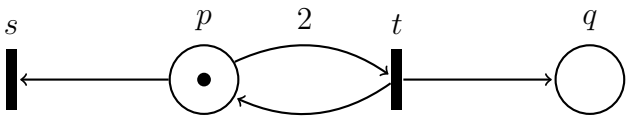


Figure 5.4 – Exemple de réseau de Petri continu pour lequel les conditions de la proposition 5.13 ne sont pas suffisantes pour témoigner de la non accessibilité.

pas accessible à partir de $\boldsymbol{m}_0$. En effet, l'unique solution à $\boldsymbol{m} = \boldsymbol{m}_0 + \mathbf{Incid} \cdot \boldsymbol{v}$ est $\boldsymbol{v} = (0,1)$, or il est impossible d'activer entièrement t en $\boldsymbol{m}_0$. L'unique piège de $\mathcal{N}$

est $\{q\}$ et il n'est pas marqué par $\boldsymbol{m}_0$. Ainsi, $\mathcal{N}, \boldsymbol{m}_0$ et $\boldsymbol{m}$ satisfont les conditions de la proposition 5.13.

Supposons que les conditions (i) et (iii) du théorème 5.4 soient satisfaites. Forcément, elles sont satisfaites par $\boldsymbol{v} = (0, 1)$. Ainsi, $\{t\} = [\![\boldsymbol{v}]\!] \in \mathsf{Activ}_{\mathcal{N}^{-1}}(\boldsymbol{m})$. Par [FH15, proposition 18], $\mathcal{N}^{-1}_{\{t\}}$ ne possède aucun siphon marqué par $\boldsymbol{m}$. Or, $\{q\}$ est un siphon de $\mathcal{N}^{-1}_{\{t\}}$ marqué par $\boldsymbol{m}$. Il y a donc une contradiction, et les conditions (i) et (iii) du théorème 5.4 ne sont pas satisfaites. □

Les deux propositions précédentes montrent que notre utilisation de l'algorithme de Fraca & Haddad et de la formule $\Phi_{\mathcal{N}}$ à l'algorithme 5.5 est déjà plus forte que l'approche de Esparza *et al.* [ELM+14]. De plus, cela apporte une justification théorique au constat que la procédure de [ELM+14] est parfois efficace en pratique ; leurs conditions sont strictement impliquées par les conditions mises au point par Fraca & Haddad [FH15] pour traiter le cas des réseaux de Petri continus.

5.5 QCover : une implémentation de l'algorithme arrière modulo accessibilité continue

Dans cette section, nous discutons d'une implémentation de l'algorithme arrière modulo accessibilité continue, et nous montrons qu'elle est efficace en comparant ses performances à celles d'autres outils.

5.5.1 Choix d'implémentation

Nous avons implémenté l'algorithme 5.5 dans un outil nommé QCover[5] à l'aide du langage de programmation Python. Ce langage a essentiellement été choisi pour sa facilité de développement et les multiples librairies disponibles. Les réseaux de Petri sont représentés naturellement en mémoire par leurs matrices **Pre** et **Post** grâce à la librairie NumPy[6]. Une représentation par matrices creuses est également supportée pour les réseaux de Petri de très grandes tailles, grâce à la

[5] QCover est disponible à `http://www-etud.iro.umontreal.ca/~blondimi/qcover/`.
[6] NumPy est disponible à `http://www.numpy.org/`.

librairie SciPy[7]. Le format d'entrée de QCover est celui de l'outil MIST[8]. Ce format permet de représenter des machines à compteurs plus générales que les réseaux de Petri et de traiter des questions plus générales que la couverture, ainsi nous ne supportons qu'un fragment de ce format.

Notre implémentation de l'algorithme de Fraca & Haddad utilise un solveur SMT afin de vérifier la satisfaisabilité des systèmes d'équations à la ligne 6 de l'algorithme 5.4. Ainsi, bien que l'algorithme de Fraca & Haddad soit polynomial, notre implémentation est exponentielle dans le pire cas. Ce choix s'appuie principalement sur deux observations. Premièrement, les meilleurs solveurs de programmation linéaire implémentent typiquement l'algorithme du simplexe, or cet algorithme n'est pas polynomial [PS98]. Deuxièmement, la quasi totalité des implémentations académiques et industrielles de l'algorithme du simplexe sont numériques, autrement dit, elles effectuent leurs calculs en arithmétique en virgule flottante. Or, afin de vérifier formellement la sûreté de systèmes modélisés par des réseaux de Petri, nous ne pouvons pas nous permettre d'obtenir des résultats erronés. Il existe néanmoins des implémentations symboliques de l'algorithme du simplexe telles que QSopt-Exact[9], parue dans [ACDE07] et détaillée dans la thèse de doctorat de Daniel Espinoza [Esp06]. Une implémentation préliminaire de l'algorithme 5.4 [Ant15] faisant l'usage de QSopt-Exact s'avérait toutefois plus lente que notre implémentation utilisant un solveur SMT.

Afin de vérifier la satisfaisabilité des formules du premier ordre, nous utilisons également un solveur SMT et nous interprétons les formules dans $\mathbb{N}$ plutôt que dans $\mathbb{Q}_{\geq 0}$ afin d'augmenter la quantité de marquages éliminés, tel que formalisé au corollaire 5.9.

Pour l'heuristique $\mathsf{min}_{c,k}$ discutée à la remarque 5.12, nous avons choisi empiriquement les paramètres $c = 10$ et $k = 5$. Nous notons que sur les instances sur lesquelles nous avons testé QCover, tous les choix raisonnables de c et k accélèrent son exécution en moyenne, bien qu'ils ralentissent également son exécution

[7] SciPy est disponible à http://www.scipy.org/scipylib/.
[8] Voir https://github.com/pierreganty/mist/wiki#input-format-of-mist.
[9] QSopt-Exact est disponible à http://www.dii.uchile.cl/~daespino/ESolver_doc/.

sur certaines instances.

5.5.2 Choix du solveur SMT

Le solveur SMT utilisé pour vérifier la satisfaisabilité des formules du premier ordre est z3 [MB08], un logiciel libre[10] de pointe développé par Microsoft Research. Le choix de ce solveur est principalement motivé par sa facilité d'utilisation, sa disponibilité dans plusieurs langages de programmation, dont Python, et ses excellentes performances. À ce titre, notons que lors de son entrée en 2008 à SMT-COMP, une compétition internationale de solveurs SMT, z3 a remporté 9 premières places et 6 deuxièmes places sur les 15 épreuves [WHM09] ; et qu'il surpasse toujours les résultats globaux obtenus par son meilleur compétiteur CVC4 à l'édition 2015 de SMT-COMP [11].

5.5.3 Test de l'outil

Nous avons évalué QCover sur 176 systèmes modélisés par des réseaux de Petri et des instances du problème de couverture, et utilisés par Esparza *et al.* afin d'évaluer leur outil Petrinizer [ELM+14]. Cette collection de systèmes est divisée en cinq catégories :

- `mist`, constituée de 27 systèmes tirés de la littérature (systèmes de production concurrents, protocoles d'exclusion mutuelle, protocoles de communication, etc.) et utilisés, entre autres, afin de tester l'outil de vérification MIST[12] ;
- `bfc`, constituée de 46 systèmes qui proviennent de programmes concurrents écrits dans le langage C (programmes utilisant des fils d'exécution et de la mémoire partagée, générateurs de nombres pseudo-aléatoires, algorithmes d'exclusion mutuelle, etc.) et qui ont été utilisés afin de tester l'outil de vérification BFC[13] [KKW12, KKW14] ;

[10] z3 est disponible à `https://z3.codeplex.com/`.
[11] Voir `http://smtcomp.sourceforge.net/2015/results-competition-main.shtml`.
[12] Voir `https://github.com/pierreganty/mist/wiki`.
[13] Voir `http://www.cprover.org/bfc/`.

- `soter`, constituée de 50 systèmes qui proviennent de programmes concurrents écrits dans le langage ERLANG et qui ont été utilisés dans [DKO13] afin de tester l'outil de vérification SOTER [DKO12] basé sur BFC ;

- `medical`, constituée de 12 systèmes, décrits dans [MMW13], modélisant l'analyse de provenance de messages d'un système médical simple de l'hôpital de l'Université Vanderbilt [BMDS07] ;

- `bug_tracking`, constituée de 41 systèmes, décrits dans [MMW13], modélisant l'analyse de provenance de messages d'un système de suivi de bogues [Jan09].

Chaque système $\mathcal{S}$ est constitué d'un réseau de Petri $\mathcal{N} = (P, T, \mathbf{Pre}, \mathbf{Post})$ et d'une instance d'une variante généralisée du problème de couverture. Une telle instance est décrite par des vecteurs $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subseteq \mathbb{N}^P$, $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n\} \subseteq \mathbb{N}^P$, et des ensembles de places $\{I_1, I_2, \ldots, I_n\} \subseteq 2^P$ et $\{E_1, E_2, \ldots, E_n\} \subseteq 2^P$. Le système $\mathcal{S}$ est dit *non sûr* s'il existe une solution à

$$\bigvee_{i=1}^{n} \left(\exists \boldsymbol{x}', \boldsymbol{y}' \in \mathbb{N}^P : \bigwedge_{p \in I_i} (\boldsymbol{x}'(p) \geq \boldsymbol{x}_i) \wedge \bigwedge_{p \in E_i} (\boldsymbol{x}'(p) = \boldsymbol{x}_i) \wedge (\boldsymbol{y}' \geq \boldsymbol{y}_i) \wedge (\boldsymbol{x}' \rightarrow^* \boldsymbol{y}') \right) .$$

Autrement, $\mathcal{S}$ est dit *sûr*. Ce vocabulaire correspond au fait que l'existence d'une solution représente une erreur, un bogue ou un état incohérent dans le système original modélisé par le réseau de Petri $\mathcal{N}$. Pour chaque système $\mathcal{S}$, nous cherchons à déterminer s'il est sûr ou non. Tel que détaillé aux tableaux 5.I et 5.II, environ les deux tiers des systèmes sont sûrs, et la taille moyenne des réseaux de Petri est de 1054 places et 8458 transitions.

Bien que le problème qui consiste à déterminer la sûreté d'un système $\mathcal{S}$ puisse sembler plus complexe que le problème de couverture, ce n'est pas le cas. En effet, nous pouvons modifier $\mathcal{S}$ de la façon suivante. Pour tout $i \in [n]$, nous ajoutons $|E_i|$ transitions au réseau de Petri $\mathcal{N}$ qui permettent chacune d'incrémenter une place $p \in E_i$ dans le but de simuler la contrainte $\boldsymbol{x}'(p) \geq \boldsymbol{x}_i(p)$. Nous obtenons ainsi le

Catégorie	Sûrs	Non sûrs	Total
`mist`	23	4	27
`bfc`	2	44	46
`soter`	38	12	50
`medical`	12	0	12
`bug_tracking`	40	1	41
Total	115	61	176

Tableau 5.I – Nombre de systèmes sûrs et non sûrs pour chaque catégorie.

système équivalent

$$\bigvee_{i=1}^{n} \left(\exists \boldsymbol{y}' \in \mathbb{N}^P : \boldsymbol{y}' \geq \boldsymbol{y}_i \wedge \boldsymbol{x}' \to^* \boldsymbol{y}' \right) \ .$$

qui correspond à résoudre n instances du problème de couverture. Pour la grande majorité des 176 systèmes, n vaut simplement 1.

Catégorie	Nombre moyen de places	Nombre moyen de transitions
`mist`	42,85	68,81
`bfc`	207,04	1 350,37
`soter`	2 804,64	4 984,58
`medical`	312	5 431
`bug_tracking`	754	27 370
Tous les systèmes	1 054,38	8 457,65

Tableau 5.II – Nombre moyen de places et de transitions dans les réseaux de Petri des systèmes pour chaque catégorie.

5.5.4 Évaluation des performances

Afin d'évaluer les performances de QCover, nous avons exécuté QCover et trois autres outils sur les 176 systèmes présentés plus haut, avec un délai d'exécution maximal de 2000 secondes (33 minutes et 20 secondes) par système, et donc de 352 000 secondes (4 jours, 1 heure, 46 minutes et 40 secondes) au total. Nous montrons que notre approche est compétitive avec les approches connues. En particulier,

QCover prouve la sûreté de plus d'instances, et ce en moins de temps, que les autres outils considérés. Dans les délais imposés, notre algorithme prouve la sûreté ou la non sûreté de 142 instances sur 176, alors que son meilleur compétiteur en prouve 122. Nous discuterons de ces résultats plus en détails. Nous démontrerons également la puissance de l'élimination de marquages en analysant le pourcentage de marquages éliminés aux différentes itérations de notre algorithme.

Nous avons comparé QCover avec les outils suivants : Petrinizer [ELM+14], MIST [Gan02] et BFC [KKW12, KKW14] dans leurs versions les plus récentes au moment d'écrire cette thèse. MIST implémente plusieurs algorithmes, nous avons utilisé son implémentation de l'algorithme arrière qui utilise une technique d'élagage basée sur les invariants de place[14] [GMD+07]. Tous les tests ont été effectués sur le même ordinateur muni de quatre processeurs Intel® Core™ 2.00 GHz i7-4510U, 8 Go de mémoire vive et Ubuntu Linux 14.04 (64 bits). Le temps d'exécution de chaque outil, sur chaque instance, a été déterminé en utilisant la somme des délais `user` et `sys` rapportés par l'outil `time` de Linux.

Catégorie	QCover	Petrinizer	MIST	BFC	Total
mist	**23**	20	22	20	23
soter	**37**	**37**	0	19	38
bfc	**2**	**2**	**2**	**2**	2
medical	**11**	4	**11**	3	12
bug_tracking	**32**	**32**	0	19	40
Total	**105**	95	35	63	115

Catégorie	QCover	Petrinizer	MIST	BFC	Total
mist	3	—	**4**	**4**	4
soter	8	—	6	**12**	12
bfc	26	—	29	**42**	44
medical	—	—	—	—	0
bug_tracking	0	—	0	**1**	1
Total	37	0	39	**59**	61

Catégorie	QCover	Petrinizer	MIST	BFC	Total
mist	**26**	20	**26**	24	27
bfc	28	2	31	**44**	46
soter	**45**	37	6	31	50
medical	**11**	4	**11**	3	12
bug_tracking	**32**	**32**	0	20	41
Total	**142**	95	74	122	176

Tableau 5.III – Nombre de systèmes sûrs (coin supérieur gauche), de systèmes non sûrs (coin supérieur droite) et nombre de systèmes décidés par chaque outil. Les nombres qui apparaissent en gras indiquent quel(s) outil(s) décide(nt) la sûreté ou non sûreté du plus grand nombre de systèmes dans leur catégorie respective.

Le nombre d'instances prouvées sûres ou non sûres par les différents outils est illustré au tableau 5.III. QCover surpasse les autres outils au chapitre des ins-

[14] Voir l'algorithme nommé `backward` à https://github.com/pierreganty/mist/wiki#coverability-checkers-included-in-mist.

tances sûres. Cela est dû à la couverture continue qui permet souvent de témoigner de la non couverture ou d'éliminer plusieurs nouveaux marquages de l'algorithme arrière. Bien que QCover demeure compétitif au chapitre des instances non sûres, BFC est l'outil qui prouve le plus d'instances. En effet, cet outil contient des heuristiques plus efficaces afin de prouver la non sûreté. Néanmoins, QCover est l'outil qui prouve le plus d'instances au total, avec 142 sur 176 instances. Notons que Petrinizer n'est pas complet et ne permet pas de prouver la non sûreté.

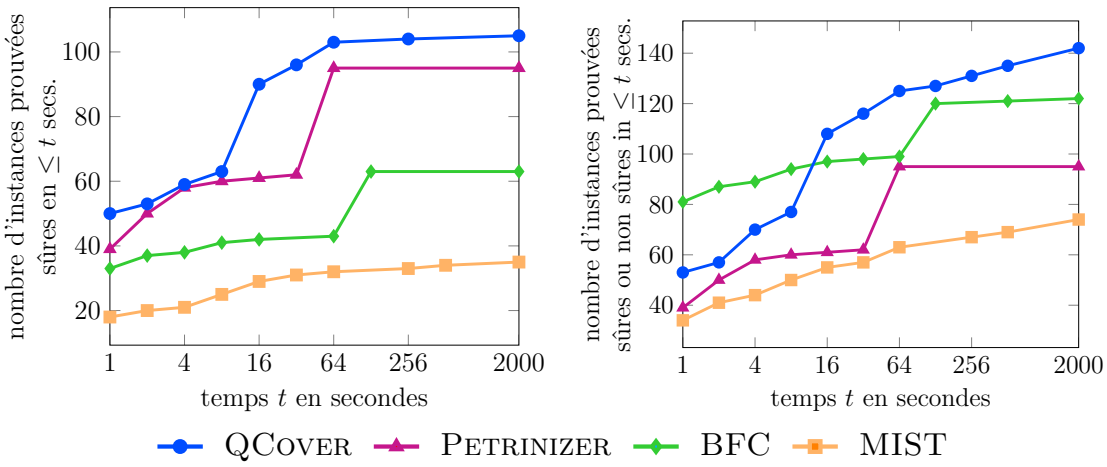


Figure 5.5 – Nombre cumulatif d'instances prouvées sûres (gauche), et d'instances prouvées sûres ou non sûres (droite) dans un délai de temps donné.

QCover ne prouve pas seulement la sûreté ou la non sûreté de plus d'instances que ses compétiteurs, il est aussi souvent plus rapide. La figure 5.5 contient deux graphiques illustrant le nombre d'instances prouvées sûres, et d'instances prouvées sûres ou non sûres, par chaque outil dans un certain délai d'exécution. Dans le cas de la sûreté, QCover surpasse les autres outils. Dans le cas général, BFC possède l'avantage sur les courts délais d'exécution. Cet avantage se manifeste principalement sur les instances relativement petites. Dès que les instances plus larges entrent en jeu, QCover surpasse BFC. Au-delà des différentes heuristiques implémentées par ces deux outils, BFC profite probablement, au niveau de la vitesse d'exécution, du choix de langage C, contre Python dans le cas de QCover.

En particulier, BFC peut décider un nombre non négligeable d'instances en moins de 10 millisecondes, ce que QCover n'accomplit sur aucune instance.

Finalement, nous considérons l'efficacité de l'accessibilité continue comme critère d'élimination de marquages. À la figure 5.6, nous illustrons le nombre de fois où un certain pourcentage de nouveaux marquages sont éliminés grâce à l'accessibilité continue, au total des itérations de l'ensemble des 176 systèmes. Étonnamment, dans un grand nombre d'itérations, plus de 95% des nouveaux marquages sont éliminés. En moyenne, 56% des nouveaux marquages sont éliminés, ce qui démontre la pertinence et l'efficacité de l'accessibilité continue comme critère d'élimination.

Mentionnons également que 83 instances peuvent être montrées sûres en vérifiant seulement la condition (i) du théorème 5.4, et que 101 instances peuvent être montrées sûres en vérifiant les trois conditions du théorème. Cela explique pourquoi QCover performe aussi bien sur les systèmes sûrs. Notons également que l'utilisation de l'algorithme de Fraca & Haddad, plutôt que de la formule $\Phi_{\mathcal{N}}$, afin d'éliminer les marquages, ne nous permet que de prouver la sûreté ou non sûreté de 132 instances plutôt que 142. De plus, l'interprétation des formules dans $\mathbb{Q}$ plutôt que dans $\mathbb{N}$ ne donne aucun gain de temps d'exécution significatif.

L'annexe B contient plus de détails et de données concernant l'évaluation de QCover.

5.6 Discussion

Dans ce chapitre, nous avons étudié le problème de couverture pour les réseaux de Petri, ainsi qu'une récente caractérisation de la relation d'accessibilité dans les réseaux de Petri continus. Nous avons mis au point un nouvel algorithme résolvant le problème de couverture pour les réseaux de Petri. Cet algorithme, nommé algorithme arrière modulo accessibilité continue, est une adaptation de l'algorithme arrière qui utilise l'accessibilité continue comme heuristique afin d'éliminer des marquages inutiles et ainsi d'accélérer les calculs. L'un des ingrédients clefs de notre heuristique provient de la formulation de l'accessibilité continue dans le fragment

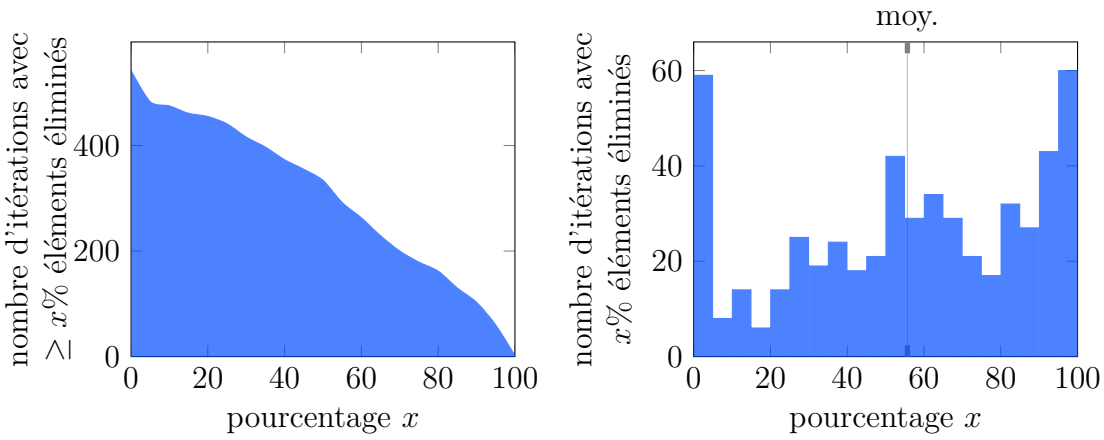


Figure 5.6 – Nombre de fois où un certain pourcentage de marquages sont éliminés grâce à l'accessibilité continue.

existentiel de $\mathsf{FO}(\mathbb{Q}_{\geq 0}, +, <)$. Nous avons montré que notre approche est plus générale que l'approche récente d'Esparza *et al.* [ELM+14]. Finalement, nous avons implémenté notre algorithme dans un outil nommé QCover et nous avons montré qu'il surpasse significativement les autres outils existants sur des systèmes tirés de la littérature.

Dans le futur, il serait prometteur de combiner notre approche à une procédure avant qui construit incrémentalement l'arbre de Karp & Miller à la façon de BFC [KKW14]. En particulier, des techniques récentes de minimisation et d'accélération dans la construction d'arbres de Karp & Miller pourraient s'avérer bénéfiques [GRB10, RS13, VH14]. Une autre façon d'améliorer empiriquement l'efficacité de QCover consisterait à utiliser des structures de données internes plus efficaces telles que les « sharing trees » [DRB04].

6

CONCLUSION

6.1 Bilan

Dans cette thèse, nous avons étudié la décidabilité et la complexité de problèmes de vérification formelle dans deux modèles de machines à compteurs : les systèmes d'addition de vecteurs et les réseaux de Petri. Nous avons également étudié une abstraction de ces machines, les systèmes de transitions bien structurés.

Dans le cas des systèmes de transitions bien structurés, nous avons étendu la notion de complétion afin de couvrir les systèmes de transitions bien structurés à branchement infini. En dépeignant les relations entre un système de transitions bien structurés et sa complétion, nous avons pu délimiter la frontière de la décidabilité de six problèmes de vérification : le problème de terminaison et sa variante forte, le problème de finitude, le problème de maintenabilité et sa version faible, et le problème de couverture. Nous avons entre autres montré qu'il est possible de décider la couverture, dans certaines classes de systèmes de transitions à branchement infini, en procédant par une approche avant plutôt qu'arrière.

Du côté des systèmes d'addition de vecteurs, nous avons établi la PSPACE-complétude du problème d'accessibilité dans le cas à deux compteurs. Pour ce faire, nous avons analysé la taille des chemins minimaux en montrant que les exécutions de tout système d'addition de vecteurs à deux compteurs peuvent être capturées par un ensemble fini de schémas linéaires de taille exponentielle et avec peu de cycles.

Nous avons également remarqué que les problèmes de finitude et de couverture sont PSPACE-complets. Au passage, nous avons établi quelques résultats de complexité nouveaux lorsque la relation d'accessibilité n'est pas restreinte aux valeurs non négatives ou lorsque les entiers sont représentés en notation unaire.

Finalement, nous avons étudié le problème de couverture dans les réseaux de Petri. Nous avons montré comment exprimer la relation d'accessibilité d'un réseau de Petri continu dans une formule de $\mathsf{FO}(\mathbb{Q}_{\geq 0}, +, <)$. Cela nous a permis d'adapter une implémentation de l'algorithme arrière pour le problème de couverture en ajoutant une élimination de marquages basée sur l'algorithme de Fraca & Haddad et notre caractérisation logique. Notre algorithme, nommé algorithme arrière modulo accessibilité continue, a été implémenté dans un outil nommé QCover. Ce dernier s'est avéré particulièrement performant et compétitif par rapport aux outils existants.

6.2 Perspectives

Nous faisons maintenant une synthèse des questions ouvertes les plus importantes et nous suggérons de nouvelles pistes de recherche.

Les résultats du chapitre 3 indiquent que les classes de WSTS effectives sous complétion sont robustes. Il serait intéressant d'identifier une telle classe générique de WSTS à branchement infini pour laquelle la notion d'arbres de Karp & Miller peut être retrouvée. Nous proposons également d'investiguer l'implémentabilité de l'algorithme de couverture avant présenté au chapitre 3 dans des modèles existants. Bien que le calcul de successeurs soit généralement plus efficace que le calcul de prédecesseurs, des heuristiques devront être mises au point afin d'énumérer efficacement des invariants inductifs. Il a récemment été remarqué que les systèmes de transitions ordonnés par un préordre sans antichaînes infinies (mais pas nécessairement un *beau* préordre) partagent des similitudes avec les WSTS. En particulier, notre algorithme pour le problème de couverture fonctionne toujours pour ces système, contrairement à l'algorithme arrière. Cette généralisation des WSTS ouvre la voie à une étude détaillée des complétions associées à des ordres sans antichaînes infinies et à une étude de la décidabilité des différents problèmes de décision.

Les techniques utilisées afin d'établir la PSPACE-complétude du problème d'accessibilité dans les VASS à deux compteurs pourraient permettre d'obtenir des

bornes de complexité dans le cas des 2-VASS étendus. Par exemple, le problème d'accessibilité est décidable pour les 2-VASS avec remises à zéro et transferts, ou dont l'un des compteurs peut être testé à zéro [FS00]. Puisque ces extensions possèdent des ensembles d'accessibilité semi-linéaires [FS00], il est envisageable de classifier la complexité de leur problème d'accessibilité grâce à des aplatissements. D'un point de vue plus appliqué, nous souhaitons continuer à mettre au point des heuristiques qui permettent de décider l'accessibilité rapidement en pratique dans les 2-VASS et leurs extensions. Dans le but de vérifier des systèmes plus complexes, nous voulons également d'étudier les VASS dont seuls deux compteurs peuvent être simultanément non bornés. Nous conjecturons que ces VASS possèdent des ensembles d'accessibilité semi-linéaires. Nous comptons aussi aborder des problèmes de synthèse pour les 2-VASS : p. ex. un VASS paramétré est un VASS dont les transitions peuvent être étiquetées par des variables ; le problème d'accessibilité consiste alors à déterminer s'il existe une affectation de ces variables qui permet d'atteindre une certaine configuration. Ce problème est dans NEXPTIME [Lec15] pour les 1-VASS paramétrés, alors que sa décidabilité est inconnue pour deux compteurs.

En ce qui concerne le problème de couverture dans les réseaux de Petri, nous souhaitons travailler sur QCover afin qu'il soit capable d'analyser les réseaux de Petri avec des arcs de transfert et de remise à zéro. Ces extensions pourraient être gérées grâce aux réseaux de Petri continus. En s'inspirant du fonctionnement de l'outil BFC [KKW14], les performances de QCover peuvent certainement être améliorées en combinant l'algorithme arrière à une approche avant permettant de trouver plus rapidement un témoin de couverture. L'utilisation de meilleures structures de données telles que les « sharing trees » [DRB04] s'avère également prometteuse pour améliorer l'efficacité de QCover.

BIBLIOGRAPHIE

[ABJ98] Abdulla, Parosh Aziz, Ahmed Bouajjani et Bengt Jonsson: *On-the-fly analysis of systems with unbounded, lossy FIFO channels*. Dans *Proc. 10^{th} International Conference on Computer Aided Verification (CAV)*, pages 305–318. Springer, 1998.

[ACBJ04] Abdulla, Parosh Aziz, Aurore Collomb-Annichini, Ahmed Bouajjani et Bengt Jonsson: *Using forward reachability analysis for verification of lossy channel systems*. Formal Methods in System Design, 25(1):39–65, 2004.

[ACDE07] Applegate, David, William J. Cook, Sanjeeb Dash et Daniel G. Espinoza: *Exact solutions to linear programming problems*. Operations Research Letters, 35(6):693–699, 2007.

[ACJT96] Abdulla, Parosh Aziz, Karlis Cerans, Bengt Jonsson et Yih-Kuen Tsay: *General decidability theorems for infinite-state systems*. Dans *Proc. 11^{th} Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 313–321. IEEE Computer Society, 1996.

[ACJT00] Abdulla, Parosh Aziz, Karlis Cerans, Bengt Jonsson et Yih-Kuen Tsay: *Algorithmic analysis of programs with well quasi-ordered domains*. Information and Computation, 160(1-2):109–127, 2000.

[AL78] Arnold, André et Michel Latteux: *Récursivité et cônes rationnels fermés par intersection*. Calcolo, 15(4):381–394, 1978.

[Ant15] Antaki, Vincent: *Accessibilité dans les systèmes à compteurs continus: théorique et pratique*. Projet honor de baccalauréat, Université de Montréal, 2015.

[BCR01] Ball, Thomas, Sagar Chaki et Sriram K. Rajamani: *Parameterized verification of multithreaded software libraries*. Dans *Proc. 7^{th} Interna-*

tional Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pages 158–173. Springer, 2001.

[BFG+15] Blondin, Michael, Alain Finkel, Stefan Göller, Christoph Haase et Pierre McKenzie: *Reachability in two-dimensional vector addition systems with states is PSPACE-complete.* Dans *Proc. 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 32–43, 2015.

[BFHH16] Blondin, Michael, Alain Finkel, Christoph Haase et Serge Haddad: *Approaching the coverability problem continuously.* Dans *Proc. 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 480–496. Springer, 2016.

[BFM14] Blondin, Michael, Alain Finkel et Pierre McKenzie: *Handling infinitely branching WSTS.* Dans *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 13–25. Springer, 2014.

[BFM16] Blondin, Michael, Alain Finkel et Pierre McKenzie: *Handling infinitely branching well-structured transition systems.* Information and Computation, à paraître, 2016.

[BG11] Bozzelli, Laura et Pierre Ganty: *Complexity analysis of the backward coverability algorithm for VASS.* Dans *Proc. 5th International Workshop on Reachability Problems (RP)*, pages 96–109. Springer, 2011.

[BMDS07] Barth, Adam, John C. Mitchell, Anupam Datta et Sharada Sundaram: *Privacy and utility in business processes.* Dans *Proc. 20th IEEE Computer Security Foundations Symposium (CSF)*, pages 279–294. IEEE Computer Society, 2007.

[BMO+12] Bouyer, Patricia, Nicolas Markey, Joël Ouaknine, Philippe Schnoebelen et James Worrell: *On termination and invariance for faulty channel machines*. Formal Aspects of Computing, 24(4-6):595–607, 2012.

[Bon75] Bonnet, R.: *On the cardinality of the set of initial intervals of a partially ordered set*. pages 189–198, 1975.

[CFI96] Cécé, Gérard, Alain Finkel et S. Purushothaman Iyer: *Unreliable channels are easier to verify than perfect channels*. Information and Computation, 124(1):20–31, 1996.

[CH16] Chistikov, Dmitry et Christoph Haase: *The taming of the semi-linear set*. Dans *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*. Leibniz-Zentrum für Informatik, à paraître, 2016. Disponible à http://www.lsv.ens-cachan.fr/~haase/documents/ch16.pdf.

[Cha88] Chan, Tat-hung: *The boundedness problem for three-dimensional vector addition systems with states*. Information Processing Letters, 26(6):287–289, 1988.

[CLM76] Cardoza, E., Richard J. Lipton et Albert R. Meyer: *Exponential space complete problems for Petri nets and commutative semigroups: Preliminary report*. Dans *Symposium on Theory of Computing (STOC)*, pages 50–54, 1976.

[Coo72] Cooper, David C.: *Theorem proving in arithmetic without multiplication*. Machine Intelligence, 7(91–99):300, 1972.

[CSMS10] Cabasino, Maria Paola, Carla Seatzu, Cristian Mahulea et Manuel Silva: *Fault diagnosis of manufacturing systems using continuous Petri nets*. Dans *Proc. IEEE International Conference on Systems, Man and Cybernetics*, pages 534–539. IEEE, 2010.

[DA87] David, René et Hassane Alla: *Continuous petri nets*. Dans *Proc. 8th European Workshop on Application and Theory of Petri nets*, tome 340, pages 275–294, 1987.

[DA10] David, René et Hassane Alla: *Discrete, Continuous, and Hybrid Petri nets*. Springer, 2^{e} édition, 2010.

[DFPS98] Dufourd, Catherine, Alain Finkel et Philippe Schnoebelen: *Reset nets between decidability and undecidability*. Dans *Proc. 25th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 103–115. Springer, 1998.

[Dic13] Dickson, Leonard Eugene: *Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors*. American Journal of Mathematics, 35(4):413–422, 1913.

[Dij87] Dijkstra, Edsger W. Rapport technique EWD1000, The University of Texas at Austin, 1987. Disponible à https://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1000.html.

[DJPS99] Dufourd, Catherine, Petr Jančar et Philippe Schnoebelen: *Boundedness of reset P/T nets*. Dans *Proc. 26th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 301–310. Springer, 1999.

[DKO12] D'Osualdo, Emanuele, Jonathan Kochems et Luke Ong: *Soter: an automatic safety verifier for Erlang*. Dans *Proc. 2nd International Workshop on Programming based on Actors, Agents, and Decentralized Control (AGERE)*, pages 137–140. ACM, 2012.

[DKO13] D'Osualdo, Emanuele, Jonathan Kochems et C.-H. Luke Ong: *Automatic verification of Erlang-style concurrency*. Dans *Proc. 20th International Symposium on Static Analysis (SAS)*, pages 454–476. Springer, 2013.

[DRB04] Delzanno, Giorgio, Jean-François Raskin et Laurent Van Begin: *Covering sharing trees: a compact data structure for parameterized verification*. International Journal on Software Tools for Technology Transfer (STTT), 5(2-3):268–297, 2004.

[ELM+14] Esparza, Javier, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp J. Meyer et Filip Niksic: *An SMT-based approach to coverability analysis*. Dans *Proc. 26th International Conference on Computer Aided Verification (CAV)*, pages 603–619. Springer, 2014.

[ELT16] Englert, Matthias, Ranko Lazić et Patrick Totzke: *Reachability in two-dimensional unary vector addition systems with states is NL-complete*. Dans *Proc. 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 2016.

[EM00] Esparza, Javier et Stephan Melzer: *Verification of safety properties using integer programming: Beyond the state equation*. Formal Methods in System Design, 16(2):159–189, 2000.

[EN98] Emerson, E. Allen et Kedar S. Namjoshi: *On model checking for non-deterministic infinite-state systems*. Dans *Proc. 13th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 70–80. IEEE Computer Society, 1998.

[End01] Enderton, Herbert B.: *A Mathematical Introduction to Logic*. Academic Press, 2^{e} édition, 2001.

[Esp06] Espinoza, Daniel G.: *On Linear Programming, Integer Programming and Cutting Planes*. Thèse de doctorat, Georgia Institute of Technology, 2006.

[ET43] Erdős, P. et A. Tarski: *On families of mutually exclusive sets*. Annals of Mathematics, 2(44):315–329, 1943.

[FG09] Finkel, Alain et Jean Goubault-Larrecq: *Forward analysis for WSTS, part I: completions.* Dans *Proc.* 26^{th} *International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 433–444. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2009.

[FG12] Finkel, Alain et Jean Goubault-Larrecq: *Forward analysis for WSTS, part II: complete WSTS.* Logical Methods in Computer Science, 8(3), 2012.

[FH15] Fraca, Estíbaliz et Serge Haddad: *Complexity analysis of continuous Petri nets.* Fundamenta Informaticae, 137(1):1–28, 2015.

[Fin87a] Finkel, Alain: *A generalization of the procedure of Karp and Miller to well structured transition systems.* Dans *Proc.* 14^{th} *International Colloquium on Automata (ICALP)*, pages 499–508. Springer, 1987.

[Fin87b] Finkel, Alain: *Well structured transitions systems.* Rapport technique 365, Université Paris-Sud, 1987.

[Fin90] Finkel, Alain: *Reduction and covering of infinite reachability trees.* Information and Computation, 89(2):144–179, 1990.

[FJ15] Fearnley, John et Marcin Jurdziński: *Reachability in two-clock timed automata is PSPACE-complete.* Information and Computation, 243:26–36, 2015.

[FL15] Finkel, Alain et Jérôme Leroux: *Recent and simple algorithms for Petri nets.* Software and System Modeling, 14(2):719–725, 2015.

[FMP04] Finkel, Alain, Pierre McKenzie et Claudine Picaronny: *A well-structured framework for analysing Petri net extensions.* Information and Computation, 195(1-2):1–29, 2004.

[FPS01] Finkel, Alain et Philippe Schnoebelen: *Well-structured transition systems everywhere!* Theoretical Computer Science, 256(1-2):63–92, 2001.

[FR75] Ferrante, Jeanne et Charles Rackoff: *A decision procedure for the first order theory of real addition with order.* SIAM Journal on Computing, 4(1):69–76, 1975.

[Fra86] Fraïssé, R.: *Theory of relations.* Studies in Logic and the Foundations of Mathematics, 118:1–456, 1986.

[Fra97] Franchella, Miriam: *On the origins of Dénes König's infinity lemma.* Archive for History of Exact Sciences, 51(1):3–27, 1997.

[FS00] Finkel, Alain et Grégoire Sutre: *Decidability of reachability problems for classes of two counters automata.* Dans *Proc.* 17^{th} *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 346–357. Springer, 2000.

[Gan02] Ganty, Pierre: *Algorithmes et structures de données efficaces pour la manipulation de contraintes sur les intervalles.* Mémoire de maîtrise, Université Libre de Bruxelles, 2002.

[GGRB09] Ganty, Pierre, Gilles Geeraerts, Jean-François Raskin et Laurent Van Begin: *Le problème de couverture pour les réseaux de Petri: résultats classiques et développements récents.* Technique et Science Informatiques, 28(9):1107–1142, 2009.

[GHPR15] Geeraerts, Gilles, Alexander Heußner, M. Praveen et Jean-François Raskin: *ω-Petri nets: Algorithms and complexity.* Fundamenta Informaticae, 137(1):29–60, 2015.

[GJ79] Garey, Michael R. et David S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1979.

[GL13] Goubault-Larrecq, Jean. communication personnelle, octobre 2013.

[GLKK16] Goubault-Larrecq, J., P. Karandikar et K. Narayan Kumar: *The ideal approach to computing closed subsets in well-quasi-orderings*, 2016. en préparation.

[GMD+07] Ganty, Pierre, Cédric Meuter, Giorgio Delzanno, Gabriel Kalyon, Jean François Raskin et Laurent Van Begin: *Symbolic data structure for sets of k-uples of integers*. Rapport technique 570, Université Libre de Bruxelles, 2007.

[GRB06a] Ganty, Pierre, Jean-François Raskin et Laurent Van Begin: *A complete abstract interpretation framework for coverability properties of WSTS*. Dans *Proc. 7th International Conference Verification, Model Checking, and Abstract Interpretation (VMCAI)*, pages 49–64. Springer, 2006.

[GRB06b] Geeraerts, Gilles, Jean-François Raskin et Laurent Van Begin: *Expand, enlarge and check: New algorithms for the coverability problem of WSTS*. Journal of Computer and System Sciences, 72(1):180–203, 2006.

[GRB10] Geeraerts, Gilles, Jean-François Raskin et Laurent Van Begin: *On the efficient computation of the minimal coverability set of Petri nets*. International Journal of Foundations of Computer Science, 21(2):135–165, 2010.

[GS66] Ginsburg, Seymour et Edwin H. Spanier: *Semigroups, Presburger formulas, and languages*. Pacific Journal of Mathematics, 16(2):285–296, 1966.

[GS92] German, Steven M. et A. Prasad Sistla: *Reasoning about systems with many processes*. Journal of the ACM, 39(3):675–735, 1992.

[Haa12] Haase, Christoph: *On the Complexity of Model Checking Counter Automata*. Thèse de doctorat, University of Oxford, 2012.

[Hac76] Hack, Michel Henri Théodore: *Decidability questions for Petri Nets.* Thèse de doctorat, Massachusetts Institute of Technology, 1976.

[HGD08] Heiner, Monika, David R. Gilbert et Robin Donaldson: *Petri nets for systems and synthetic biology.* Dans *Formal Methods for Computational Systems Biology. Proc. 8^{th} International School on Formal Methods for the Design of Computer*, pages 215–264. Springer, 2008.

[HH14] Haase, Christoph et Simon Halfon: *Integer vector addition systems with states.* Dans *Proc. 8^{th} International Workshop on Reachability Problems (RP)*, tome 8762 de *Lecture Notes in Computer Science*, pages 112–124. Springer, 2014.

[Hig52] Higman, Graham: *Ordering by divisibility in abstract algebras.* Proceedings of the London Mathematical Society, s3-2(1):326–336, 1952.

[HKOW09] Haase, Christoph, Stephan Kreutzer, Joël Ouaknine et James Worrell: *Reachability in succinct and parametric one-counter automata.* Dans *Proc. 20^{th} International Conference on Concurrency Theory (CONCUR)*, pages 369–383. Springer, 2009.

[HMU01] Hopcroft, John E., Rajeev Motwani et Jeffrey D. Ullman: *Introduction to automata theory, languages, and computation.* Addison-Wesley-Longman, $2^{\text{ème}}$ édition, 2001.

[Hoa85] Hoare, C. A. R.: *Communicating Sequential Processes.* Prentice-Hall, 1985.

[HP79] Hopcroft, John E. et Jean-Jacques Pansiot: *On the reachability problem for 5-dimensional vector addition systems.* Theoretical Computer Science, 8:135–159, 1979.

[HRHY86] Howell, Rodney R., Louis E. Rosier, Dung T. Huynh et Hsu-Chun Yen: *Some complexity bounds for problems concerning finite and 2-*

dimensional vector addition systems with states. Theoretical Computer Science, 46(3):107–140, 1986.

[Jan99] Jančar, Petr: *A note on well quasi-orderings for powersets.* Information Processing Letters, 72(5-6):155–160, 1999.

[Jan09] Janák, Jiří: *Issue tracking systems.* Mémoire de maîtrise, Masarykova univerzita (Université Masaryk), 2009.

[JRS03] Júlvez, Jorge, Laura Recalde et Manuel Silva: *On reachability in autonomous continuous Petri net systems.* Dans *Proc. 24th International Conference on Applications and Theory of Petri Nets (ICATPN)*, pages 221–240. Springer, 2003.

[KKW12] Kaiser, Alexander, Daniel Kroening et Thomas Wahl: *Efficient coverability analysis by proof minimization.* Dans *Proc. 23rd International Conference on Concurrency Theory (CONCUR)*, pages 500–515. Springer, 2012.

[KKW14] Kaiser, Alexander, Daniel Kroening et Thomas Wahl: *A widening approach to multithreaded program verification.* ACM Transactions on Programming Languages and Systems, 36(4):14:1–14:29, 2014.

[KM67] Karp, Richard M. et Raymond E. Miller: *Parallel program schemata: a mathematical model for parallel computation.* Dans *Proc. 8th Annual Symposium on Switching and Automata Theory*, pages 55–61. IEEE Computer Society, 1967.

[Kos82] Kosaraju, S. Rao: *Decidability of reachability in vector addition systems (preliminary version).* Dans *Proc. 14th Annual ACM Symposium on Theory of Computing (STOC)*, pages 267–281. ACM, 1982.

[KPS96] Kouchnarenko, Olga et Philippe Schnoebelen: *A model for recursive-parallel programs.* Electronic Notes in Theoretical Computer Science, 5:30, 1996.

[Lam92] Lambert, Jean-Luc: *A structure to decide reachability in Petri nets.* Theoretical Computer Science, 99(1):79–104, 1992.

[Lec15] Lechner, Antonia: *Synthesis problems for one-counter automata.* Dans *Proc. 9^{th} International Workshop on Reachability Problems (RP)*, pages 89–100. Springer, 2015.

[Ler09] Leroux, Jérôme: *The general vector addition system reachability problem by presburger inductive invariants.* Dans *Proc. 24^{th} Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 4–13. IEEE Computer Society, 2009.

[Ler11] Leroux, Jérôme: *Vector addition system reachability problem: a short self-contained proof.* Dans *Proc. 5^{th} International Conference on Language and Automata Theory and Applications (LATA)*, pages 41–64. Springer, 2011.

[Ler12] Leroux, Jérôme: *Vector addition systems reachability problem (a simpler solution).* Dans *The Alan Turing Centenary Conference*, pages 214–228. EasyChair, 2012.

[Lev95] Leveson, Nancy G.: *Safeware: System Safety and Computers.* Addison-Wesley, 1995. Extrait disponible à http://sunnyday.mit.edu/papers/therac.pdf.

[Lip76] Lipton, Richard J.: *The reachability problem requires exponential space.* Rapport technique 63, Department of Computer Science, Yale University, 1976.

[Lip09] Lipton, Richard J.: *An EXPSPACE lower bound.* https://rjlipton.wordpress.com/2009/04/08/an-expspace-lower-bound/, 2009. Consulté le 7 mars 2016.

[LS04] Leroux, Jérôme et Grégoire Sutre: *On flatness for 2-dimensional vector addition systems with states*. Dans *Proc. 15th International Conference on Concurrency Theory (CONCUR)*, pages 402–416. Springer, 2004.

[LS15] Leroux, Jérôme et Sylvain Schmitz: *Demystifying reachability in vector addition systems*. Dans *Proc. 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 56–67. IEEE, 2015.

[LST15] Leroux, Jérôme, Grégoire Sutre et Patrick Totzke: *On the coverability problem for pushdown vector addition systems in one dimension*. Dans *Proc. 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 324–336. Springer, 2015.

[May81] Mayr, Ernst W.: *An algorithm for the general Petri net reachability problem*. Dans *Proc. 13th Annual ACM Symposium on Theory of Computing (STOC)*, pages 238–246. ACM, 1981.

[MB08] Mendonça de Moura, Leonardo et Nikolaj Bjørner: *Z3: an efficient SMT solver*. Dans *Proc. 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 337–340. Springer, 2008.

[MMW13] Majumdar, Rupak, Roland Meyer et Zilong Wang: *Static provenance verification for message passing programs*. Dans *Proc. 20th International Symposium on Static Analysis (SAS)*, pages 366–387. Springer, 2013.

[Pac82] Pachl, Jan K.: *Reachability problems for communicating finite state machines*. Rapport technique CS-82-12, University of Waterloo, mai 1982.

[Per14] Perifel, Sylvain: *Complexité algorithmique*. Ellipses, 2014. Disponible librement à https://www.irif.univ-paris-diderot.fr/~sperifel/complexite.pdf.

[Pet62] Petri, Carl Adam: *Kommunikation mit Automaten*. Thèse de doctorat, Universität Hamburg (Université de Hambourg), 1962.

[Pot91] Pottier, Loïc: *Minimal solutions of linear diophantine systems: bounds and algorithms*. Dans *Proc. 4^{th} International Conference on Rewriting Techniques and Applications*, pages 162–173. Springer, 1991.

[Pou04] Poulsen, Kevin: *Tracking the blackout bug*. Disponible à http://www.securityfocus.com/news/8412, avril 2004. Consulté le 19 mars 2016.

[Pre29] Presburger, Mojżesz: *Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt*. Comptes Rendus du I$^{\text{er}}$ Congrès des mathématiciens des pays slaves, pages 192–201, 1929.

[PS98] Papadimitriou, Christos H. et Ken Steiglitz: *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

[Rac78] Rackoff, Charles: *The covering and boundedness problems for vector addition systems*. Theoretical Computer Science, 6:223–231, 1978.

[Rad54] Rado, R.: *Partial well-ordering of sets of vectors*. Mathematika, 1:89–95, juin 1954.

[Reu88] Reutenauer, Christophe: *Aspects mathématiques des réseaux de Petri*. Études et recherches en informatique. Masson, Paris, 1988.

[RHB+09] Rogers, James, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome et Sean Wibel: *On languages piecewise testable in the strict sense*. Dans *Proc. 10^{th} and 11^{th} Biennial Conference on the Mathematics of Language (MOL)*, pages 255–265. Springer, 2009.

[RLM96] Reddy, V.N., M.N. Liebman et M.L. Mavrovouniotis: *Qualitative analysis of biochemical reaction systems.* Computers in biology and medicine, 26(1):9–24, 1996.

[RS13] Reynier, Pierre-Alain et Frédéric Servais: *Minimal coverability set for Petri nets: Karp and Miller algorithm with pruning.* Fundamenta Informaticae, 122(1-2):1–30, 2013.

[RTS99] Recalde, Laura, Enrique Teruel et Manuel Silva: *Autonomous continuous P/T systems.* Dans *Proc. 20th International Conference on Application and Theory of Petri Nets 1999 (ICATPN)*, pages 107–126. Springer, 1999.

[RY86] Rosier, Louis E. et Hsu Chun Yen: *A multiparameter analysis of the boundedness problem for vector addition systems.* Journal of Computer and System Sciences, 32(1):105–135, 1986.

[Sch98] Schrijver, Alexander: *Theory of linear and integer programming.* John Wiley & Sons, 1998.

[Sch16] Schmitz, Sylvain: *The complexity of reachability in vector addition systems.* ACM SIGLOG News, 3(1), 2016.

[Sip06] Sipser, Michael: *Introduction to the theory of computation.* Thomson Course Technology, 2^{e} édition, 2006.

[Son85] Sontag, Eduardo D.: *Real addition and the polynomial hierarchy.* Information Processing Letters, 20(3):115–120, 1985.

[SS12] Schmitz, Sylvain et Philippe Schnoebelen: *Algorithmic Aspects of WQO Theory.* Recueil de notes de cours, 2012.

[ST77] Sacerdote, George S. et Richard L. Tenney: *The decidability of the reachability problem for vector addition systems (preliminary version).*

Dans *Proc.* 9^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 61–76. ACM, 1977.

[To10] To, Anthony W.: *Model checking infinite-state systems: generic and specific approaches*. Thèse de doctorat, University of Edinburgh, 2010.

[van98] van der Aalst, Wil M.P.: *The application of Petri nets to workflow management*. Journal of circuits, systems, and computers, 8(1):21–66, 1998.

[VH14] Valmari, Antti et Henri Hansen: *Old and new algorithms for minimal coverability sets*. Fundamenta Informaticae, 131(1):1–25, 2014.

[VP75] Valiant, Leslie G. et Mike Paterson: *Deterministic one-counter automata*. Journal of Computer and System Sciences, 10(3):340–350, 1975.

[VSS05] Verma, Kumar Neeraj, Helmut Seidl et Thomas Schwentick: *On the complexity of equational horn clauses*. Dans *Proc.* 20^{th} *International Conference on Automated Deduction (CADE)*, pages 337–352. Springer, 2005.

[WHM09] Wintersteiger, Christoph M., Youssef Hamadi et Leonardo Mendonça de Moura: *A concurrent portfolio approach to SMT solving*. Dans *Proc.* 21^{st} *International Conference on Computer Aided Verification (CAV)*, pages 715–720. Springer, 2009.

[ZWH12] Zufferey, Damien, Thomas Wies et Thomas A. Henzinger: *Ideal abstractions for well-structured transition systems*. Dans *Proc.* 13^{th} *International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, 2012.

A

PREUVE DU LEMME 4.12

Dans cette annexe, nous démontrons le lemme 4.12. Nous donnons dans un premier temps une série de propositions concernant les ensembles semi-linéaires.

A.1 Propositions préliminaires

Proposition A.1 ([CH16])**.** *Soient* $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{Z}^d$ *et* $Q \subseteq R \subseteq \mathbb{Z}^d$ *finis. Il existe* $B \subseteq \mathbb{Z}^d$ *fini tel que* $\|B\| \leq (|R| \cdot \max(\|\boldsymbol{c}\|, \|\boldsymbol{d}\|, \|R\|))^{O(d)}$ *et*

$$\mathit{Lin}(\boldsymbol{c}, Q) \cap \mathit{Lin}(\boldsymbol{d}, R) = \bigcup_{\boldsymbol{b} \in B} \mathit{Lin}(\boldsymbol{b}, Q) \ .$$

Rappelons que $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{Q}^2$ sont linéairement indépendants si, et seulement si, $\boldsymbol{p}(1)\boldsymbol{q}(2) \neq \boldsymbol{q}(1)\boldsymbol{p}(2)$.

Proposition A.2 (Règle de Cramer)**.** *Soient* $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{c} \in \mathbb{Q}^2$ *tels que* $\boldsymbol{p}$ *et* $\boldsymbol{q}$ *sont linéairement indépendants. L'unique solution* $\lambda_1, \lambda_2 \in \mathbb{Q}$ *au système d'équations linéaires* $\lambda_1 \boldsymbol{p} + \lambda_2 \boldsymbol{q} = \boldsymbol{c}$ *est*

$$\begin{aligned} \lambda_1 &= \frac{\boldsymbol{c}(1)\boldsymbol{q}(2) - \boldsymbol{q}(1)\boldsymbol{c}(2)}{\boldsymbol{p}(1)\boldsymbol{q}(2) - \boldsymbol{q}(1)\boldsymbol{p}(2)} \\ \lambda_2 &= \frac{\boldsymbol{p}(1)\boldsymbol{c}(2) - \boldsymbol{c}(1)\boldsymbol{p}(2)}{\boldsymbol{p}(1)\boldsymbol{q}(2) - \boldsymbol{q}(1)\boldsymbol{p}(2)} \ . \end{aligned}$$

Proposition A.3. *Soient* $\boldsymbol{b} \in \mathbb{Z}^d$ *et* $P \subseteq \mathbb{Z}^d$ *fini. Il existe* $C \subseteq \mathbb{Z}^d$ *fini tel que* $\|C\| \leq \|\boldsymbol{b}\| + |P| \cdot \|P\|$ *et*

$$\mathit{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P) \cap \mathbb{N}^d = \bigcup_{\boldsymbol{c} \in C} \mathit{Lin}(\boldsymbol{c}, P) \ .$$

Démonstration. Posons $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$. Soit $\boldsymbol{v} \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{c}, P) \cap \mathbb{N}^d$, il existe $\lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{Q}_{\geq 0}$ tels que $\boldsymbol{v} = \boldsymbol{b} + \lambda_1 \boldsymbol{p}_1 + \lambda_2 \boldsymbol{p}_2 + \cdots + \lambda_n \boldsymbol{p}_n$. Pour tout $i \in [n]$, posons $\lambda'_i \stackrel{\text{déf}}{=} \lambda_i - \lfloor \lambda_i \rfloor$. Nous obtenons

$$\boldsymbol{v} = \underbrace{\boldsymbol{b} + \lambda'_1 \boldsymbol{p}_1 + \lambda'_2 \boldsymbol{p}_2 + \cdots + \lambda'_n \boldsymbol{p}_n}_{\boldsymbol{c}} + \lfloor \lambda_1 \rfloor \boldsymbol{p}_1 + \lfloor \lambda_2 \rfloor \boldsymbol{p}_2 + \cdots + \lfloor \lambda_n \rfloor \boldsymbol{p}_n \ .$$

Puisque $\boldsymbol{v} \in \mathbb{N}^d$, forcément $\boldsymbol{c} \in \mathbb{N}^d$ et ainsi $\boldsymbol{v} \in \mathsf{Lin}(\boldsymbol{c}, P)$. L'ensemble $C \stackrel{\text{déf}}{=} \mathsf{Lin}_{\{0 \leq a < 1 : a \in \mathbb{Q}_{\geq 0}\}}(\boldsymbol{b}, P) \cap \mathbb{N}^d$ satisfait donc la proposition puisque $\|C\| \leq \|\boldsymbol{b}\| + |P| \cdot \|P\|$. □

Proposition A.4. *Soient $P \subseteq \mathbb{Z}^2$ et $\boldsymbol{b} \in P$,*

(i) s'il existe $a' \in \mathbb{Q}_{>0}$ tel que $(a', 0) \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P)$, alors il existe $a \in \mathbb{N}$ tel que $a \leq \|P\|^{O(1)}$ et $(a, 0) \in \mathsf{Lin}(\boldsymbol{b}, P)$;

(ii) s'il existe $b' \in \mathbb{Q}_{>0}$ tel que $(0, b') \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P)$, alors il existe $b \in \mathbb{N}$ tel que $b \leq \|P\|^{O(1)}$ et $(0, b) \in \mathsf{Lin}(\boldsymbol{b}, P)$.

Démonstration. Nous montrons le point (i), le point (ii) est symétrique. Soit $a' \in \mathbb{Q}_{>0}$ tel que $(a', 0) \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P)$. Posons $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$. Nous pouvons supposer, sans perte de généralité, que $\boldsymbol{p}_1 = \boldsymbol{b}$. Il existe donc $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{N}$ et $\beta_1, \beta_2, \ldots, \beta_n \in \mathbb{N}_{>0}$ tels que

$$(a', 0) = \left(1 + \frac{\alpha_1}{\beta_1}\right) \boldsymbol{p}_1 + \frac{\alpha_2}{\beta_2} \boldsymbol{p}_2 + \cdots + \frac{\alpha_n}{\beta_n} \boldsymbol{p}_n \ . \tag{A.1}$$

En multipliant les deux côtés de (A.1) par $\beta = \beta_1 \beta_2 \cdots \beta_n$, nous obtenons

$$(a'\beta, 0) = (\beta + \alpha_1 \beta_2 \beta_3 \cdots \beta_n) \boldsymbol{p}_1 + (\alpha_2 \beta_1 \beta_3 \cdots \beta_n) \boldsymbol{p}_2 + \cdots + (\alpha_n \beta_1 \beta_2 \cdots \beta_{n-1}) \boldsymbol{p}_n \ .$$

Ainsi, $(a'\beta, 0) \in \mathsf{Lin}(\boldsymbol{b}, P)$ et $a'\beta \in \mathbb{N}_{>0}$, mais $a'\beta$ n'est toujours pas borné. Consi-

dérons le système d'équations diophantiennes linéaires $\mathcal{E}$ suivant :

$$\mathcal{E} : \begin{bmatrix} \boldsymbol{p}_1 & \boldsymbol{p}_2 & \cdots & \boldsymbol{p}_n & \begin{pmatrix} -1 \\ 0 \end{pmatrix} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ y \end{bmatrix} = -\boldsymbol{b}$$

Puisque $(a'\beta, 0) \in \mathsf{Lin}(\boldsymbol{b}, P)$, $\mathcal{E}$ possède une solution. Ainsi, par le corollaire 4.17, $\mathcal{E}$ possède une solution $\boldsymbol{e} \in \mathbb{N}^{n+1}$ telle que $\|\boldsymbol{e}\| \leq (2 \cdot \max(\|\boldsymbol{p}_1\|, \|\boldsymbol{p}_2\|, \ldots, \|\boldsymbol{p}_n\|, 1) + \|\boldsymbol{b}\|)^{O(1)}$. Puisque $\boldsymbol{b} \in P$, nous avons $\|\boldsymbol{e}\| \leq \|P\|^{O(1)}$. Ainsi, $a \stackrel{\text{déf}}{=} \boldsymbol{e}(n+1)$ est tel que recherché. □

A.2 Preuve du lemme

Nous prouvons, à la proposition A.5, le même énoncé que celui du lemme 4.12 mais en bornant la norme des vecteurs tirés de $\mathsf{Lin}(\boldsymbol{b}, P)$ plutôt que la taille des coefficients des combinaisons linéaires. La preuve du lemme 4.12 en découlera ensuite aisément.

Proposition A.5. *Soient $P \subseteq \mathbb{Z}^2$ et $\boldsymbol{b} \in P$. Pour tout quadrant Z, il existe $m \in \mathbb{N}_{>0}$ et*

$$\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_m \in Z \cap \mathit{Lin}(\boldsymbol{b}, P)$$
$$P_1, P_2, \ldots, P_m \subseteq Z \cap [\mathit{Lin}(\boldsymbol{b}, P) \cup P]$$

tels que pour tout $i \in [m]$, $|P_i| \leq 2$, $\|\boldsymbol{c}_i\|, \|P_i\| \leq (|P| \cdot \|P\|)^{O(1)}$ et

$$\mathit{Lin}(\boldsymbol{b}, P) \cap Z = \bigcup_{i \in [m]} \mathit{Lin}(\boldsymbol{c}_i, P_i) \ .$$

Démonstration. Nous démontrons le lemme seulement pour $Z = \mathbb{N}^2$, les trois autres cas étant symétriques. Nous divisons $\mathsf{Lin}(\boldsymbol{b}, P) \cap \mathbb{N}^2$ en un nombre fini de régions

tel qu'illustré à la figure A.1. Posons

$$L_{\mathsf{inf}} \stackrel{\text{déf}}{=} \mathsf{Lin}(\boldsymbol{b}, P) \cap \{(a,b) \in \mathbb{N}^2 : a \geq \max(0, \boldsymbol{b}(1)),\ b \geq \max(0, \boldsymbol{b}(2))\}\ ,$$

$$L_i^{\updownarrow} \stackrel{\text{déf}}{=} \mathsf{Lin}(\boldsymbol{b}, P) \cap \{(a,b) \in \mathbb{N}^2 : a = i,\ b \geq 0\}\ ,$$

$$L_j^{\leftrightarrow} \stackrel{\text{déf}}{=} \mathsf{Lin}(\boldsymbol{b}, P) \cap \{(a,b) \in \mathbb{N}^2 : a \geq 0,\ b = j\}\ .$$

L'ensemble $\mathsf{Lin}(\boldsymbol{b}, P) \cap \mathbb{N}^2$ se réécrit de la façon suivante :

$$\mathsf{Lin}(\boldsymbol{b}, P) \cap \mathbb{N}^2 = \bigcup_{0 \leq i < \boldsymbol{b}(1)} L_i^{\updownarrow} \cup \bigcup_{0 \leq j < \boldsymbol{b}(2)} L_j^{\leftrightarrow} \cup L_{\mathsf{inf}}\ .$$

Afin de démontrer que $\mathsf{Lin}(\boldsymbol{b}, P) \cap \mathbb{N}^2 = \bigcup_{i \in [m]} \mathsf{Lin}(\boldsymbol{c}_i, P_i)$ pour des vecteurs appropriés, nous montrons comment obtenir de tels ensembles semi-linéaires pour chacune des régions. Nous supposons sans perte de généralité que $\mathbf{0} \notin P$.

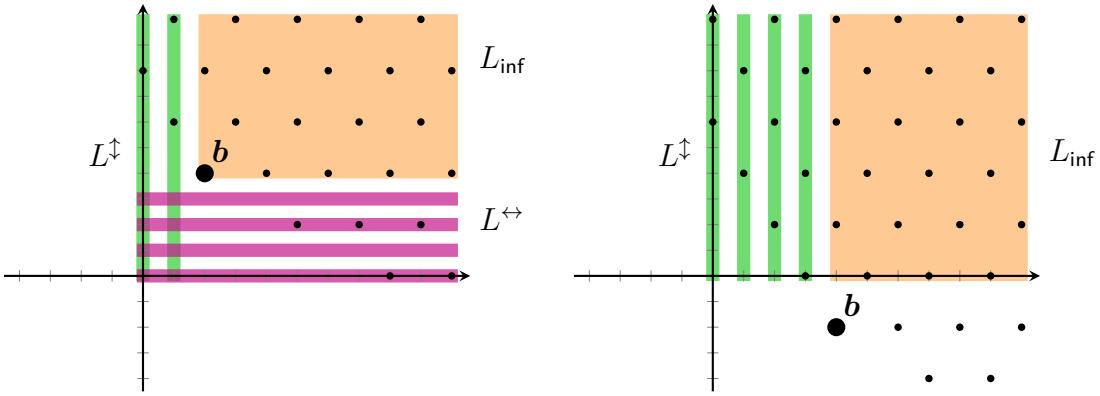


Figure A.1 – Exemples de la décomposition de $\mathsf{Lin}(\boldsymbol{b}, P) \cap \mathbb{N}^2$ où $L^{\updownarrow}$, $L^{\leftrightarrow}$ et L_{inf} sont les points compris dans les zones colorées respectivement en ■, ■ et ■. *Gauche :* $\boldsymbol{b} = (2,4)$ et $P = \{(-1,2),(3,-2),\boldsymbol{b}\}$; *droite :* $\boldsymbol{b} = (4,-2)$ et $P = \{(-1,2),(3,-2),\boldsymbol{b}\}$.

Région $L_i^{\updownarrow}$: Soit $0 \leq i < \boldsymbol{b}(1)$. S'il n'existe pas de vecteur $(i,j) \in \mathsf{Lin}(\boldsymbol{b}, P)$ situé

sur la verticale $\{i\} \times \mathbb{Z}$ au-dessus de la ligne de l'origine à $\boldsymbol{b}$, c.-à-d., tel que

$$j > \frac{\boldsymbol{b}(2)}{\boldsymbol{b}(1)} \cdot i \ , \tag{A.2}$$

alors $L_i^{\updownarrow}$ est fini. Nous avons donc terminé puisque $L_i^{\updownarrow} = \bigcup_{\boldsymbol{c} \in L_i^{\updownarrow}} \mathsf{Lin}(\boldsymbol{c}, \emptyset)$ et $\|L_i^{\updownarrow}\| \leq \|\boldsymbol{b}\| \leq \|P\|$. Supposons donc qu'il existe $(i,j) \in \mathsf{Lin}(\boldsymbol{b}, P)$ qui satisfait (A.2). Posons $\boldsymbol{v} \stackrel{\text{déf}}{=} (i,j) - \boldsymbol{b}$ et $\lambda = -\boldsymbol{b}(1)/\boldsymbol{v}(1)$. Nous obtenons,

$$\boldsymbol{b} + \lambda \boldsymbol{v} = (0, \underbrace{(\boldsymbol{b}(1)j - \boldsymbol{b}(2)i)/(\boldsymbol{b}(1) - i)}_{b'}) \ .$$

Par (A.2) et $\boldsymbol{b}(1) > i \geq 0$, nous avons $\lambda \geq 0$ et $b' > 0$. Ainsi, $(0, b') \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P)$, puisque $\boldsymbol{v} \in \mathsf{Cone}(P)$. Par la proposition A.4, il existe donc $b \in \mathbb{N}_{>0}$ tel que $b \leq \|P\|^{O(1)}$ et $(0,b) \in \mathsf{Lin}(\boldsymbol{b}, P)$. En particulier, $(0,b) \in \mathsf{Cone}(P)$ car $\boldsymbol{b} \in P$, et conséquemment,

$$\begin{aligned}
L_i^{\updownarrow} &= \mathsf{Lin}(\boldsymbol{b}, P) \cap (\{i\} \times \mathbb{N}) \\
&= \mathsf{Lin}(\boldsymbol{b}, P) \cap \bigcup_{0 \leq j < b} \mathsf{Lin}((i,j), \{(0,b)\}) \\
&= \bigcup_{0 \leq j < b} \mathsf{Lin}(\boldsymbol{b}, P) \cap \mathsf{Lin}((i,j), \{(0,b)\}) \\
&= \bigcup_{0 \leq j < b} \mathsf{Lin}(\boldsymbol{b}, P \cup \{(0,b)\}) \cap \mathsf{Lin}((i,j), \{(0,b)\}) \qquad (\text{car } (0,b) \in \mathsf{Cone}(P)) \\
&= \bigcup_{0 \leq j < b} \bigcup_{\boldsymbol{c} \in C_j} \mathsf{Lin}(\boldsymbol{c}, \{(0,b)\}) \qquad (\text{par la proposition A.1})
\end{aligned}$$

où $C_j \subseteq \mathbb{Z}^2$ et $\|C_j\| \leq ((|P|+1) \cdot \max(\boldsymbol{b}, \max(i,j), \|P\|, b))^{O(1)} \leq (|P| \cdot \|P\|)^{O(1)}$.

Région $L_j^{\leftrightarrow}$: Symétrique au cas précédent en obtenant $a \in \mathbb{N}_{>0}$ tel que $(a,0) \in \mathsf{Lin}(\boldsymbol{b}, P)$.

Région L_{inf} : Par le théorème de Carathéodory [Sch98, p. 94], pour tout $\boldsymbol{v} \in \mathsf{Lin}(\boldsymbol{b}, P)$, il existe un ensemble de vecteurs linéairement indépendants $P' \subseteq P$

tel que $\boldsymbol{v} \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P')$ et forcément, en dimension deux, $|P'| \leq 2$. Ainsi,

$$L_{\mathsf{inf}} = \bigcup_{\substack{P' \subseteq P \\ |P'| \leq 2,\ P' \text{ lin. ind.}}} \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap L_{\mathsf{inf}} \ .$$

Afin de terminer la preuve, il suffit donc de considérer $\mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap L_{\mathsf{inf}}$ pour un certain P'. Fixons ainsi un ensemble $P' \subseteq P$ de vecteurs linéairement indépendants tel que $|P'| \leq 2$. Posons $\wr\boldsymbol{b} \stackrel{\text{déf}}{=} \{\boldsymbol{b}' \in \mathbb{Q}^2 : \boldsymbol{b}' \geq \boldsymbol{b}\}$. Nous montrerons qu'il existe $Q \subseteq \mathbb{N}^2 \cap [\mathsf{Lin}(\boldsymbol{b}, P) \cup P]$ tel que $|Q| \leq 2$, $\|Q\| \leq \|P\|^{O(1)}$ et

$$\mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \ \cap \ \wr\boldsymbol{b} = \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, Q) \ . \tag{A.3}$$

L'existence d'un tel Q complète la preuve :

$$\begin{aligned}
\mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap L_{\mathsf{inf}} &= \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap \wr\boldsymbol{b} \cap \mathbb{N}^2 \cap \mathsf{Lin}(\boldsymbol{b}, P) && \text{(par déf. de } L_{\mathsf{inf}}) \\
&= \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, Q) \cap \mathbb{N}^2 \cap \mathsf{Lin}(\boldsymbol{b}, P) && \text{(par (A.3))} \\
&= \bigcup_{c \in C} \mathsf{Lin}(\boldsymbol{c}, Q) \cap \mathsf{Lin}(\boldsymbol{b}, P) && (C \subseteq \mathbb{Z}^2 \text{ par prop. A.3}) \\
&= \bigcup_{c \in C} \mathsf{Lin}(\boldsymbol{c}, Q) \cap \mathsf{Lin}(\boldsymbol{b}, P \cup Q) && (Q \subseteq \underbrace{\mathsf{Lin}(\boldsymbol{b}, P)}_{\subseteq\ \mathsf{Cone}(P) \text{ car } \boldsymbol{b} \in P}) \\
&= \bigcup_{c \in C} \bigcup_{d \in D_{\boldsymbol{c}}} \mathsf{Lin}(\boldsymbol{d}, Q) && (D_{\boldsymbol{c}} \subseteq \mathbb{Z}^2 \text{ par prop. A.1})
\end{aligned}$$

où

$$\begin{aligned}
\|C\| &\leq \|\boldsymbol{b}\| + |Q| \cdot \|Q\| && \leq 3\|P\|^{O(1)}, \text{ et} \\
\|D_{\boldsymbol{c}}\| &\leq ((|P| + |Q|) \cdot \max(\|\boldsymbol{b}\|, \|\boldsymbol{c}\|, \|P\|, \|Q\|))^{O(1)} && \leq (|P| \cdot \|P\|)^{O(1)} \ .
\end{aligned}$$

Il reste à prouver (A.3). Nous procédons en trois cas :

(1) $|P'| \leq 1$ ou $P' \subseteq \mathbb{N}^2$. Si $P' = \{\boldsymbol{p}\}$ où $\boldsymbol{p} \notin \mathbb{N}^2$, alors $\mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap \wr\boldsymbol{b} = \{\boldsymbol{b}\}$, et ainsi il suffit de poser $Q \stackrel{\text{déf}}{=} \emptyset$. Autrement, $P' \subseteq \mathbb{N}^2$ et nous posons $Q \stackrel{\text{déf}}{=} P'$.

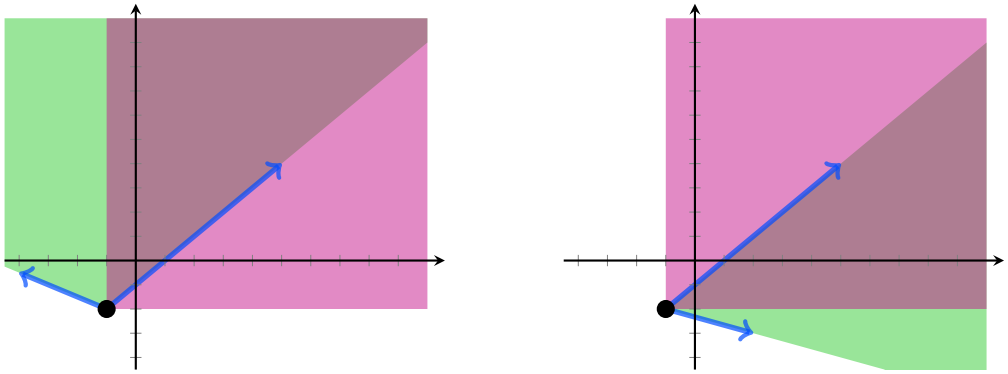


Figure A.2 – *Gauche :* exemple du cas (2–i) ; *droite :* exemple du cas (2–ii). La zone colorée en ▬ représente $\mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, \{\boldsymbol{p}, \boldsymbol{q}\})$, la zone colorée en ▬ représente $\wr\boldsymbol{b}$, et $\boldsymbol{p}$ et $\boldsymbol{q}$ apparaissent en ▬.

(2) $P' = \{\boldsymbol{p}, \boldsymbol{q}\}$, $\boldsymbol{p} \in \mathbb{N}^2$ et $\boldsymbol{q} \notin \mathbb{N}^2$. Puisque $\boldsymbol{p}$ et $\boldsymbol{q}$ sont linéairement indépendants, nous pouvons considérer deux cas :

(i) $\boldsymbol{p}(1)\boldsymbol{q}(2) > \boldsymbol{q}(1)\boldsymbol{p}(2)$,

(ii) $\boldsymbol{p}(1)\boldsymbol{q}(2) < \boldsymbol{q}(1)\boldsymbol{p}(2)$.

Supposons (i) et cherchons à démontrer que $(0,1) \in \mathsf{Cone}_{\mathbb{Q}_{\geq 0}}(P')$ et qu'il existe $b \in \mathbb{N}_{>0}$ tel que $(0,b) \in \mathsf{Lin}(\boldsymbol{b}, P')$. Le cas (ii) est traité symétriquement en montrant l'existence de $(1,0) \in \mathsf{Cone}_{\mathbb{Q}_{\geq 0}}(P')$ et d'un $a \in \mathbb{N}_{>0}$ tel que $(a,0) \in \mathsf{Lin}(\boldsymbol{b}, P')$.

Observons d'abord que $\boldsymbol{q}(1) < 0$. En effet, si $\boldsymbol{q}(1) \geq 0$, alors $\boldsymbol{q}(2) < 0$ car $\boldsymbol{q} \notin \mathbb{N}^2$, et ainsi $\boldsymbol{p}(1)\boldsymbol{q}(2) \leq 0 \leq \boldsymbol{q}(1)\boldsymbol{p}(2)$, ce qui est une contradiction. Si $\boldsymbol{p} = (0,b)$ pour $b \in \mathbb{N}_{>0}$, alors $\mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap \wr\boldsymbol{b} = \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, \{\boldsymbol{p}\})$ et la preuve est complétée en posant $Q \stackrel{\text{déf}}{=} \{\boldsymbol{p}\}$. Supposons donc que $\boldsymbol{p}(1) > 0$. Soit $\boldsymbol{v} \in \mathbb{Z}^2$. Puisque $\boldsymbol{p}(1) > 0$, $\boldsymbol{q}(1) < 0$ et $\boldsymbol{p}(1)\boldsymbol{q}(2) > \boldsymbol{q}(1)\boldsymbol{p}(2)$, il existe $\lambda_1, \lambda_2 \in \mathbb{Q}_{\geq 0}$ et $b' \in \mathbb{Q}_{>0}$ tels que $\boldsymbol{v} + \lambda_1\boldsymbol{p} + \lambda_2\boldsymbol{q} = (0,b')$. En plus détaillé, en choisissant

$$b' \;=\; \max\left(1, \left\lceil \frac{-\boldsymbol{v}(1)\boldsymbol{q}(2)}{\boldsymbol{q}(1)} + \boldsymbol{v}(2) \right\rceil, \left\lceil \frac{-\boldsymbol{v}(1)\boldsymbol{p}(2)}{\boldsymbol{p}(1)} + \boldsymbol{v}(2) \right\rceil\right)$$

la règle de Cramer impose

$$\begin{aligned}\lambda_1 &= \frac{-\boldsymbol{q}(1)(b' - \boldsymbol{v}(2)) - \boldsymbol{v}(1)\boldsymbol{q}(2)}{\boldsymbol{p}(1)\boldsymbol{q}(2) - \boldsymbol{q}(1)\boldsymbol{p}(2)} \\ \lambda_2 &= \frac{\boldsymbol{p}(1)(b' - \boldsymbol{v}(2)) + \boldsymbol{v}(1)\boldsymbol{p}(2)}{\boldsymbol{p}(1)\boldsymbol{q}(2) - \boldsymbol{q}(1)\boldsymbol{p}(2)} \ .\end{aligned}$$

et l'on se convainc que $\lambda_1, \lambda_2 \geq 0$. Ainsi en prenant $\boldsymbol{v} = \boldsymbol{0}$, nous obtenons $(0,1) \in \mathsf{Cone}_{\mathbb{Q}_{\geq 0}}(P')$, et en prenant $\boldsymbol{v} = \boldsymbol{b}$, nous obtenons $(0,b') \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P')$ pour un certain $b' \in \mathbb{Q}_{>0}$. Par la proposition A.4, il existe donc $b \in \mathbb{N}_{>0}$ tel que $b \leq \|P\|^{O(1)}$ et $(0,b) \in \mathsf{Lin}(\boldsymbol{b}, P)$. Nous concluons en prenant $Q \stackrel{\text{déf}}{=} \{\boldsymbol{p}, (0,b)\}$ et en montrant que $\mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, Q) = \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap \wr\boldsymbol{b}$.

$\subseteq$: Soit $\boldsymbol{v} \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, Q)$, il existe $\lambda_1, \lambda_2 \in \mathbb{Q}_{\geq 0}$ tels que $\boldsymbol{v} = \boldsymbol{b} + \lambda_1\boldsymbol{p} + \lambda_2(0,b)$, et ainsi $\boldsymbol{v} = \boldsymbol{b} + \lambda_1\boldsymbol{p} + (\lambda_2/b)(0,1) \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, \{\boldsymbol{p}, (0,1)\})$. Puisque $\boldsymbol{p} \in P' \cap \mathbb{N}^2$ et $(0,1) \in \mathsf{Cone}_{\mathbb{Q}_{\geq 0}}(P')$, nous obtenons $\boldsymbol{v} \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap \wr\boldsymbol{b}$.

$\supseteq$: Soit $\boldsymbol{v} \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap \wr\boldsymbol{b}$. Il existe $\lambda_1, \lambda_2 \in \mathbb{Q}_{\geq 0}$ tels que $\boldsymbol{v} = \lambda_1\boldsymbol{p} + \lambda_2\boldsymbol{q}$. Par la règle de Cramer, nous avons $\boldsymbol{v} = \boldsymbol{b} + \lambda_1'\boldsymbol{p} + \lambda_2'(0,b)$ où

$$\begin{aligned}\lambda_1' &= \frac{\boldsymbol{v}(1) - \boldsymbol{b}(1)}{\boldsymbol{p}(1)} \\ \lambda_2' &= \frac{\lambda_2(\boldsymbol{p}(1)\boldsymbol{q}(2) - \boldsymbol{q}(1)\boldsymbol{p}(2))}{b \cdot \boldsymbol{p}(1)} \ .\end{aligned}$$

Puisque, $\boldsymbol{v}(1) \geq \boldsymbol{b}(1)$, $\boldsymbol{p}(1) > 0$, $\lambda_2 \geq 0$, $\boldsymbol{p}(1)\boldsymbol{q}(2) > \boldsymbol{q}(1)\boldsymbol{p}(2)$ et $b > 0$, nous obtenons $\lambda_1', \lambda_2' \geq 0$. Par conséquent, $\boldsymbol{v} \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, Q)$.

(3) $\underline{P' = \{\boldsymbol{p}, \boldsymbol{q}\} \text{ et } \boldsymbol{p}, \boldsymbol{q} \notin \mathbb{N}^2}$. Notons d'abord que si $\boldsymbol{p}(1), \boldsymbol{q}(1) < 0$ ou $\boldsymbol{p}(2), \boldsymbol{q}(2) < 0$, alors $\mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap \wr\boldsymbol{b} = \{\boldsymbol{b}\}$ et nous avons ainsi terminé. Nous pouvons donc supposer, sans perte de généralité, que $\boldsymbol{p} \in \mathbb{N} \times -\mathbb{N}_{>0}$ et $\boldsymbol{q} \in -\mathbb{N}_{>0} \times \mathbb{N}$, autrement il suffit d'interchanger les vecteurs. Puisque $\boldsymbol{p}$ et $\boldsymbol{q}$ sont linéairement indépendants, deux cas sont possibles :

(i) $\boldsymbol{p}(1)\boldsymbol{q}(2) > \boldsymbol{q}(1)\boldsymbol{p}(2)$,

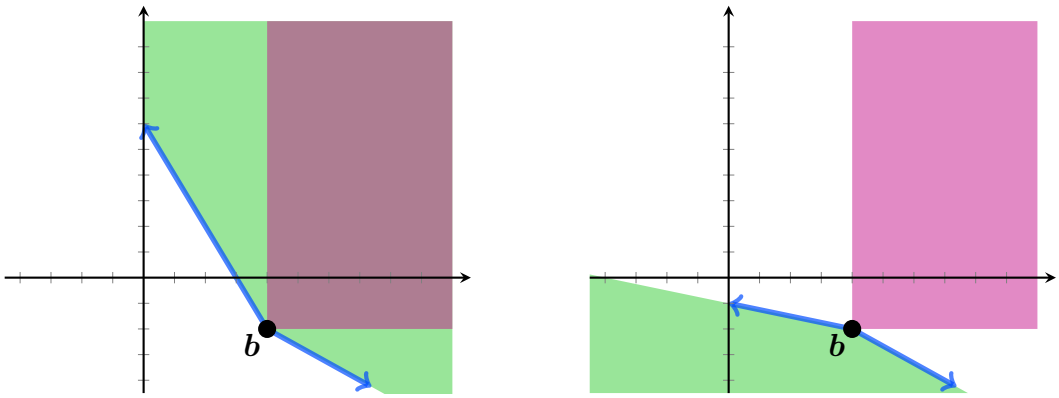


Figure A.3 – *Gauche :* exemple du cas (3–i) ; exemple du cas (3–ii). La zone colorée en ▬ représente $\mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, \{\boldsymbol{p}, \boldsymbol{q}\})$, la zone colorée en ▬ représente $\wr\boldsymbol{b}$, et $\boldsymbol{p}$ et $\boldsymbol{q}$ apparaissent en ▬.

(ii) $\boldsymbol{p}(1)\boldsymbol{q}(2) < \boldsymbol{q}(1)\boldsymbol{p}(2)$.

À l'aide de la règle de Cramer, il est possible de vérifier que le cas (ii) implique $\mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap \wr\boldsymbol{b} = \{\boldsymbol{b}\}$ (côté droit de la figure A.3). Nous supposons donc le cas (i). Remarquons que $\boldsymbol{p}(1) = 0$ ou $\boldsymbol{q}(2) = 0$ enfreint (i). Ainsi, tel qu'illustré du côté gauche de la figure A.3,

$$\boldsymbol{p}(1) > 0,\ \boldsymbol{p}(2) < 0,\ \boldsymbol{q}(1) < 0 \text{ et } \boldsymbol{q}(2) > 0\ . \tag{A.4}$$

Comme au cas (2), par (A.4), il est possible d'appliquer la règle de Cramer afin d'obtenir $(0,1), (1,0) \in \mathsf{Cone}_{\mathbb{Q}_{\geq 0}}(P')$ et $a', b' \in \mathbb{Q}_{>0}$ tels que $(a',0), (0,b') \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P')$. Par la proposition A.4, il existe donc $a, b \in \mathbb{N}_{>0}$ tels que $a, b \leq \|P\|^{O(1)}$ et $(a,0), (0,b) \in \mathsf{Lin}(\boldsymbol{b}, P)$. Posons $Q \stackrel{\text{déf}}{=} \{(a,0), (0,b)\}$. Nous concluons en prenant $Q \stackrel{\text{déf}}{=} \{(a,0), (0,b)\}$ et en montrant que $\mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, Q) = \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap \wr\boldsymbol{b}$.

$\subseteq$: Soit $\boldsymbol{v} \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, Q)$, il existe $\lambda_1, \lambda_2 \in \mathbb{Q}_{\geq 0}$ tels que $\boldsymbol{v} = \boldsymbol{b} + \lambda_1(a,0) + \lambda_2(0,b)$, et ainsi $\boldsymbol{v} = \boldsymbol{b} + a\lambda_1(1,0) + b\lambda_2(0,1) \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, \{(1,0), (0,1)\})$. Puisque $(1,0), (0,1) \in \mathsf{Cone}_{\mathbb{Q}_{\geq 0}}(P')$, nous obtenons $\boldsymbol{v} \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap \wr\boldsymbol{b}$.

$\supseteq$: Soit $\boldsymbol{v} \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, P') \cap \wr\boldsymbol{b}$, nous avons $\boldsymbol{v} = \boldsymbol{b} + (\boldsymbol{v}(1) - \boldsymbol{b}(1))/a \cdot (a,0) +$

$(\boldsymbol{v}(2) - \boldsymbol{b}(2))/b \cdot (0, b)$. Conséquemment $\boldsymbol{v} \in \mathsf{Lin}_{\mathbb{Q}_{\geq 0}}(\boldsymbol{b}, Q)$, puisque $\boldsymbol{v} \geq \boldsymbol{b}$. □

Lemme 4.12. Soient $P \subseteq \mathbb{Z}^2$ et $\boldsymbol{b} \in P$. Pour tout quadrant Z, il existe $e \leq (|P| \cdot \|P\|)^{O(1)}$, $m \in \mathbb{N}_{>0}$ et

$$\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_m \in Z \cap \mathsf{Lin}_{[0,e]}(\boldsymbol{b}, P)$$
$$P_1, P_2, \ldots, P_m \subseteq Z \cap [\mathsf{Lin}_{[0,e]}(\boldsymbol{b}, P) \cup P]$$

tels que pour tout $i \in [m]$, $|P_i| \leq 2$, et

$$\mathsf{Lin}(\boldsymbol{b}, P) \cap Z = \bigcup_{i \in [m]} \mathsf{Lin}(\boldsymbol{c}_i, P_i) \ .$$

Démonstration. Soient les vecteurs $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_m \in Z \cap \mathsf{Lin}(\boldsymbol{b}, P)$ et $P_1, P_2, \ldots, P_m \subseteq Z \cap [\mathsf{Lin}(\boldsymbol{b}, P) \cup P]$ obtenus par la proposition A.5. Nous montrons que ces vecteurs satisfont l'énoncé du lemme pour un certain $e \leq (|P| \cdot \|P\|)^{O(1)}$.

Posons

$$D \stackrel{\text{déf}}{=} \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_m\} \cup \{\boldsymbol{p} \in P_i \cap \mathsf{Lin}(\boldsymbol{b}, P) : i \in [m]\}$$

et $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$. Soit $\boldsymbol{v} \in D$. Considérons le système d'équations diophantiennes linéaires $\mathcal{E}$ suivant :

$$\mathcal{E} : \begin{bmatrix} \boldsymbol{p}_1 & \boldsymbol{p}_2 & \cdots & \boldsymbol{p}_n \end{bmatrix} \boldsymbol{x} = \boldsymbol{v} - \boldsymbol{b}$$

Puisque $\boldsymbol{v} \in \mathsf{Lin}(\boldsymbol{b}, P)$, $\mathcal{E}$ possède une solution. Ainsi, par le corollaire 4.17, il existe $e_{\boldsymbol{v}} \leq (\|P\| + \|\boldsymbol{v}\|)^{O(1)}$ et une solution $\boldsymbol{x} \in \mathbb{N}^d$ de $\mathcal{E}$ tels que $\|\boldsymbol{x}\| \leq e_{\boldsymbol{v}}$. Puisque $\|\boldsymbol{v}\| \leq (|P| \cdot \|P\|)^{O(1)}$, la preuve est complétée en prenant $e \stackrel{\text{déf}}{=} \max\{e_{\boldsymbol{v}} : \boldsymbol{v} \in D\}$. □

B

ÉVALUATION DÉTAILLÉE DE QCOVER

B.1 Sommaire des résultats de sûreté et non sûreté

Catégorie	QCover	Petrinizer	MIST	BFC	Total
mist	**23**	20	22	20	23
medical	**11**	4	**11**	3	12
bfc	**2**	**2**	**2**	**2**	2
bug_tracking	**32**	**32**	0	19	40
soter	**37**	**37**	0	19	38
Total	**105**	95	35	63	115

Tableau B.I – Nombre d'instances prouvées sûres par les différents outils.

Catégorie	QCover	Petrinizer	MIST	BFC	Total
mist	3	—	**4**	**4**	4
medical	—	—	—	—	0
bfc	26	—	29	**42**	44
bug_tracking	0	—	0	**1**	1
soter	8	—	6	**12**	12
Total	37	0	39	**59**	61

Tableau B.II – Nombre d'instances prouvées non sûres par les différents outils.

Catégorie	QCover	Petrinizer	MIST	BFC	Total
mist	**26**	20	**26**	24	27
medical	**11**	4	**11**	3	12
bfc	28	2	31	**44**	46
bug_tracking	**32**	**32**	0	20	41
soter	**45**	37	6	31	50
Total	**142**	95	74	122	176

Tableau B.III – Nombre d'instances prouvées sûres ou non sûres par les différents outils.

B.2 Temps d'exécution détaillés

Instances	QCover	Petrinizer	MIST	BFC
Instances sûres				
fms_attic	**216**	499	641	3274
bingham_h50	244	503	3845	**9**
csm	206	489	10	**7**
bingham_h25	218	496	244	**7**
bingham_h150	376	538	751718	**38**
MultiME	246	1201	**6**	**6**
pingpong	169	1190	6	**5**
mesh2x2	177	493	**120**	45358
bingham_h250	416	578	trop long	**99**
fms	175	482	**4**	114
multipool	**170**	500	304	1897
mesh3x2	**191**	494	1395	trop long
extendedread-write-smallconsts	416	—	**17**	mémoire
bingham_h250_attic	243196	**11783**	14373	13195
manufacturing	178	—	1225	**5**
extendedread-write	**414**	—	140456	mémoire
basicME	182	1191	**3**	9
lamport	183	1188	6	**5**
kanban	168	489	**3**	10
peterson	279	1897	**5**	6
read-write	167	482	25	**6**
newdekker	185	1191	9	**7**
newrtp	170	491	**4**	5
Instances non sûres				
kanban	trop long	—	432179	**81**
pncsasemiliv	2615	—	75	**7**
pncsacover	37169	—	4045	**1968**
leabasicapproach	442	—	**5**	6

Tableau B.IV – Temps d'exécution des différents outils sur la catégorie `mist`. Chaque temps d'exécution apparaît en millisecondes. Les couleurs vert (rapide) et rouge (lent) indiquent la rapidité de l'outil en comparaison au meilleur et pire temps obtenus parmi tous les outils.

Instances	QCover	Petrinizer	MIST	BFC
Instances sûres				
conditionals_vs_satabs.2	320	542	785	**156**
rand_cas_vs_satabs.2	234	505	159	**41**
Instances non sûres				
double_lock_p3_vs_satabs.1	23898	—	1772	**27**
Boop_simple_vf_satabs.1	885	—	24	**11**
conditionals_vs_satabs.1	4950	—	90	**16**
rand_lock_p0_vs_satabs.2	1608119	—	15262	**278**
double_lock_p2_vs_satabs.1	301840	—	231990	**53**
double_lock_p2_vs_satabs.2	trop long	—	trop long	**579**
szymanski_vs_satabs.2	trop long	—	trop long	**16857**
double_lock_p1_vs_satabs.2	trop long	—	trop long	**14624**
Function_Pointer3_vs_satabs.2	trop long	—	243560	**957**
rand_cas_vs_satabs.1	3307	—	116	**15**
Boop_simple_vf_satabs.2	trop long	—	trop long	**5833**
simple_loop5_vs_satabs.2	trop long	—	trop long	**216**
lu-fig2_fixed_vs_satabs.1	1965	—	79	**8**
constants_vf_satabs.2	158996	—	1139	**358**
dekker_vs_satabs.1	32629	—	1101	**15**
Function_Pointer3_vs_satabs.3	trop long	—	trop long	trop long
stack_cas_p0_vs_satabs.3	trop long	—	**697796**	trop long
buggy_spaghetti_vf_satabs.2	292369	—	2608	**631**
simple_loop5_vs_satabs.1	2869	—	139	**9**
dekker_vs_satabs.2	trop long	—	trop long	**612**
pthread5_vs_satabs.4	trop long	—	trop long	**531**
double_lock_p3_vs_satabs.2	trop long	—	trop long	**128**
constants_vf_satabs.1	442	—	7	**6**
pthread5_vs_satabs.1	1604031	—	928836	**46**
double_lock_p1_vs_satabs.3	trop long	—	trop long	**10774**
rand_lock_p0_vs_satabs.1	2672	—	150	**12**
double_lock_p1_vs_satabs.1	370721	—	59678	**27**
pthread5_vs_satabs.3	trop long	—	trop long	**562**
lu-fig2_fixed_vs_satabs.2	153145	—	46447	**21**
spin2003_vs_satabs.1	8354	—	411	**10**
rand_lock_p0_vs_satabs.3	trop long	—	**162388**	1052180
peterson_vs_satabs.1	8517	—	356	**9**
spin2003_vs_satabs.2	561779	—	657321	**55**
buggy_spaghetti_vf_satabs.1	4259	—	85	**23**
Function_Pointer3_vs_satabs.1	5257	—	249	**16**
stack_cas_p0_vs_satabs.2	248850	—	51281	**34**
stack_cas_p0_vs_satabs.1	15774	—	2528	**14**
stack_lock_p0_vs_satabs.2	trop long	—	trop long	**330**
pthread5_vs_satabs.2	1614516	—	1230991	**47**
lu-fig2_fixed_vs_satabs.3	trop long	—	trop long	**474**
stack_lock_p0_vs_satabs.1	11984	—	1083	**11**
szymanski_vs_satabs.1	560765	—	6116	**37**
peterson_vs_satabs.2	trop long	—	trop long	**2861**
double_lock_p3_vs_satabs.3	trop long	—	trop long	**1823**

Tableau B.V – Temps d'exécution des différents outils sur la catégorie `bfc`. Chaque temps d'exécution apparaît en millisecondes. Les couleurs vert (rapide) et rouge (lent) indiquent la rapidité de l'outil en comparaison au meilleur et pire temps obtenus parmi tous les outils.

Instances	QCover	Petrinizer	MIST	BFC
Instances sûres				
safe_send__sending_to_non-pid_3__depth_1	492	521	trop long	**77**
finite_leader__single_leader__depth_1	**367**	524	trop long	1279
ring__single_message_in_mailbox__depth_1	**20574**	24505	trop long	— †
parikh__should_already_be_initialized__depth_0	199	493	trop long	**13**
safe_send__sending_to_non-pid_1__depth_2	547	2015	trop long	**76**
finite_leader__single_leader__depth_2	**372**	523	trop long	1297
reslock__critical__depth_0	**318**	506	trop long	trop long
parikh__should_already_be_initialized__depth_2	201	498	trop long	**13**
safe_send__sending_to_non-pid_2__depth_1	537	2788	trop long	**92**
sieve__single_message_in_sieve_mailbox__depth_0	**283**	511	trop long	trop long
sieve__single_message_in_counter_mailbox__depth_1	1018	**725**	trop long	trop long
safe_send__sending_to_non-pid_4__depth_1	576	7304	trop long	**78**
reslockbeh__critical__depth_1	2199	**730**	trop long	trop long
concdb__single_client_writes__depth_2	5108	**1972**	trop long	trop long
reslockbeh__critical__depth_2	5740	**1050**	trop long	trop long
ring__single_message_in_mailbox__depth_2	mémoire	trop long	trop long	— †
safe_send__sending_to_non-pid_4__depth_2	652	7291	trop long	**86**
state_factory__single_message_in_mailbox__depth_1	**41838**	43668	trop long	— †
sieve__single_message_in_counter_mailbox__depth_0	**280**	517	trop long	trop long
safe_send__sending_to_non-pid_1__depth_1	539	2030	trop long	**78**
sieve__single_message_in_counter_mailbox__depth_2	2464	**1614**	trop long	trop long
parikh__should_already_be_initialized__depth_1	205	494	trop long	**15**
state_factory__after_receive_if_no_mail__depth_0	367	523	trop long	**119**
pipe__single_message_in_mailbox__depth_1	2293	**1363**	trop long	trop long
pipe__single_message_in_mailbox__depth_2	1066	**721**	trop long	trop long
pipe__single_message_in_mailbox__depth_0	252	500	trop long	**37**
reslock__critical__depth_1	**441**	545	trop long	trop long
firewall__no_pred_called_with_zero__depth_2	3043	**1205**	trop long	94843
safe_send__sending_to_non-pid_3__depth_2	307	520	trop long	**90**
concdb__single_client_writes__depth_0	**421**	532	trop long	trop long
concdb__single_client_writes__depth_1	1073	**705**	trop long	trop long
reslockbeh__critical__depth_0	**459**	529	trop long	trop long
ring__single_message_in_mailbox__depth_0	**331**	526	trop long	1506
sieve__single_message_in_filter_mailbox__depth_0	**288**	506	trop long	trop long
safe_send__sending_to_non-pid_2__depth_2	542	2788	trop long	**79**
state_factory__single_message_in_mailbox__depth_0	376	514	trop long	**128**
firewall__no_pred_called_with_zero__depth_1	**506**	542	trop long	528
reslock__critical__depth_2	**582**	585	trop long	trop long
Instances non sûres				
stutter__we_abhorr_as__depth_2	73119	—	34873	**11**
stutter__we_abhorr_as__depth_1	71319	—	34962	**12**
unsafe_send__sending_to_non-pid__depth_1	2301	—	83	**7**
firewall__no_pred_called_with_zero__depth_0	1146433	—	trop long	**81**
howait__all_workers_finished_if_wait_over__depth_2	trop long	—	trop long	**93265**
howait__all_workers_finished_if_wait_over__depth_1	trop long	—	trop long	**7293**
unsafe_send__sending_to_non-pid__depth_2	2295	—	81	**6**
howait__all_workers_finished_if_wait_over__depth_0	trop long	—	trop long	**36**
safe_send__sending_to_non-pid__depth_0	324125	—	trop long	**16**
unsafe_send__sending_to_non-pid__depth_0	2297	—	77	**7**
finite_leader__single_leader__depth_0	trop long	—	trop long	**123**
stutter__we_abhorr_as__depth_0	31512	—	1422	**12**

Tableau B.VI – Temps d'exécution des différents outils sur la catégorie `soter`. Chaque temps d'exécution apparaît en millisecondes. Les couleurs vert (rapide) et rouge (lent) indiquent la rapidité de l'outil en comparaison au meilleur et pire temps obtenus parmi tous les outils. Les trois instances marquées par le symbole † n'ont pas pu être évalués avec l'outil BFC, puisque l'outil de conversion `spec2tts` fourni par BFC afin de convertir le format `.spec` au format `.tts` a manqué de mémoire.

Instances	QCover	Petrinizer	MIST	BFC
Instances sûres				
x0_HA_q5	trop long	—	**3072158**	trop long
x0_AR_q2	41492	—	**11352**	trop long
x0_AA_q2	**40143**	—	53875	trop long
x0_HA_q2	42911	—	**11686**	trop long
x0_AA_q5	1865112	—	**453338**	trop long
x0_HQ_q5	39566	—	**20699**	trop long
x0_AR_q5	40358	—	**20014**	trop long
x0_HQ_q2	48101	—	**11811**	trop long
x0_AR_q1	2573	**2191**	7638	5101
x0_HQ_q1	2734	**2141**	7571	trop long
x0_AA_q1	4099	**2171**	7522	5010
x0_HA_q1	4637	**2147**	7609	5103

Tableau B.VII – Temps d'exécution des différents outils sur la catégorie `medical`. Chaque temps d'exécution apparaît en millisecondes. Les couleurs vert (rapide) et rouge (lent) indiquent la rapidité de l'outil en comparaison au meilleur et pire temps obtenus parmi tous les outils.

Instances	QCover	Petrinizer	MIST	BFC
Instances sûres				
x0_PENDING_q1	**18611**	46088	—	106065
x0_PENDING_q8	trop long	—	—	trop long
x0_FIXED_q1	**15710**	46105	—	trop long
x0_VERIFIED_q3	**15467**	46357	—	104123
x0_BUG_REPORT_q5	**15600**	46507	—	trop long
x0_PENDING_q3	**15409**	46795	—	104941
x0_FIXED_q5	**15462**	46159	—	104901
x0_MORE_INFO_q8	trop long	—	—	trop long
x0_FIX_AGAIN_q1	**15697**	46144	—	106723
x0_FIXED_q8	trop long	—	—	trop long
x0_MORE_INFO_q3	**15481**	46808	—	trop long
x0_BUG_REPORT_q3	**15510**	46815	—	trop long
x0_BUG_REPORT_q1	**15353**	46014	—	trop long
x0_CLOSED_q5	**15469**	46707	—	104905
x0_MORE_INFO_q5	**15419**	46506	—	trop long
x0_MUST_FIX_q5	**16039**	46351	—	trop long
x0_MUST_FIX_q6	**16063**	46113	—	105480
x0_BUG_REPORT_q8	trop long	—	—	trop long
x0_CLOSED_q8	trop long	—	—	trop long
x0_FIXED_q6	**15451**	46892	—	trop long
x0_PENDING_q5	**16055**	46738	—	107307
x0_FIXED_q3	**15427**	47159	—	trop long
x0_FIX_AGAIN_q5	**16329**	45886	—	107330
x0_VERIFIED_q1	**15466**	46645	—	106631
x0_MORE_INFO_q6	**15577**	45350	—	107455
x0_VERIFIED_q8	trop long	—	—	trop long
x0_CLOSED_q1	**15427**	45867	—	104714
x0_PENDING_q6	**15543**	46877	—	105400
x0_CLOSED_q6	**15478**	46740	—	trop long
x0_FIX_AGAIN_q6	**15592**	46306	—	104390
x0_MUST_FIX_q1	**15540**	46386	—	107811
x0_VERIFIED_q6	**15444**	46595	—	105518
x0_BUG_REPORT_q6	**15485**	47031	—	trop long
x0_FIX_AGAIN_q8	trop long	—	—	trop long
x0_FIX_AGAIN_q3	**15401**	46453	—	trop long
x0_VERIFIED_q5	**15521**	46057	—	104425
x0_MUST_FIX_q8	trop long	—	—	trop long
x0_MUST_FIX_q3	**15567**	46580	—	trop long
x0_CLOSED_q3	**15436**	46693	—	104557
x0_MORE_INFO_q1	**15418**	45929	—	105397
Instances non sûres				
x0_PENDING_q3_buggy	trop long	—	—	**287966**

Tableau B.VIII – Temps d'exécution des différents outils sur la catégorie `bug_tracking`. Chaque temps d'exécution apparaît en millisecondes. Les couleurs vert (rapide) et rouge (lent) indiquent la rapidité de l'outil en comparaison au meilleur temps obtenu parmi tous les outils. MIST n'a supporté aucun fichier de cette catégorie dû à des erreurs de mémoire.

B.3 Comparaison des temps d'exécution sur toutes les instances

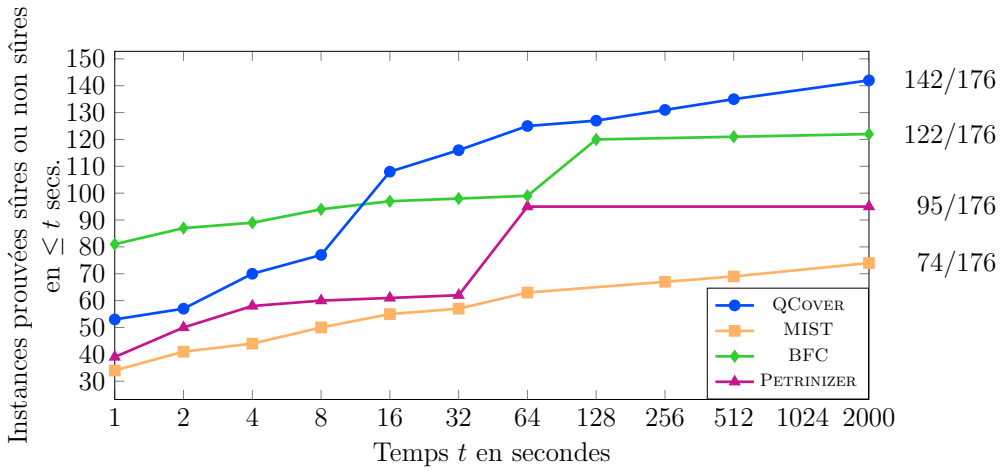


Figure B.1 – Nombre cumulatif d'instances prouvées sûres ou non sûres pour l'ensemble des catégories.

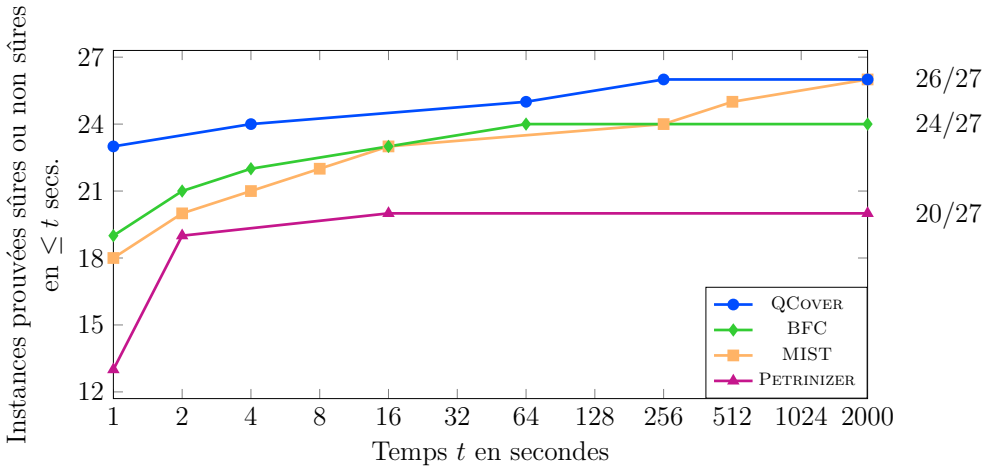


Figure B.2 – Nombre cumulatif d'instances prouvées sûres ou non sûres dans la catégorie `mist`.

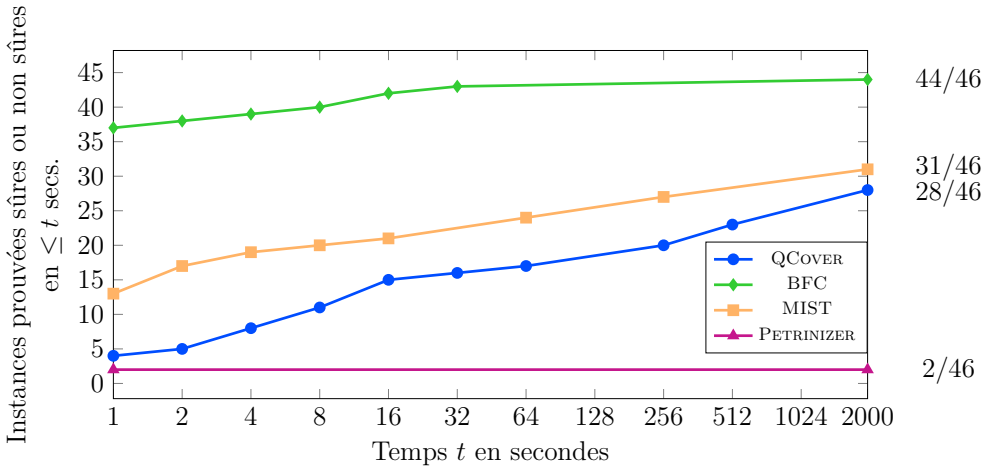


Figure B.3 – Nombre cumulatif d'instances prouvées sûres ou non sûres dans la catégorie `bfc`.

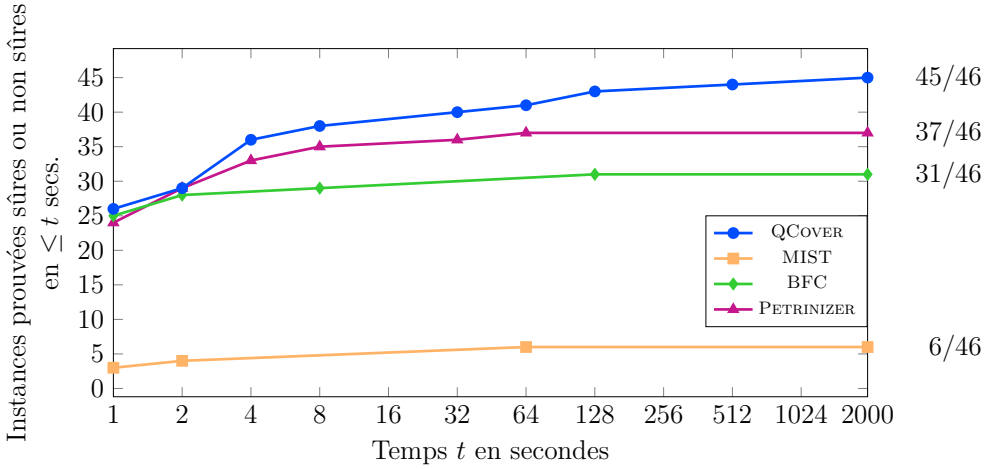


Figure B.4 – Nombre cumulatif d'instances prouvées sûres ou non sûres dans la catégorie `soter`.

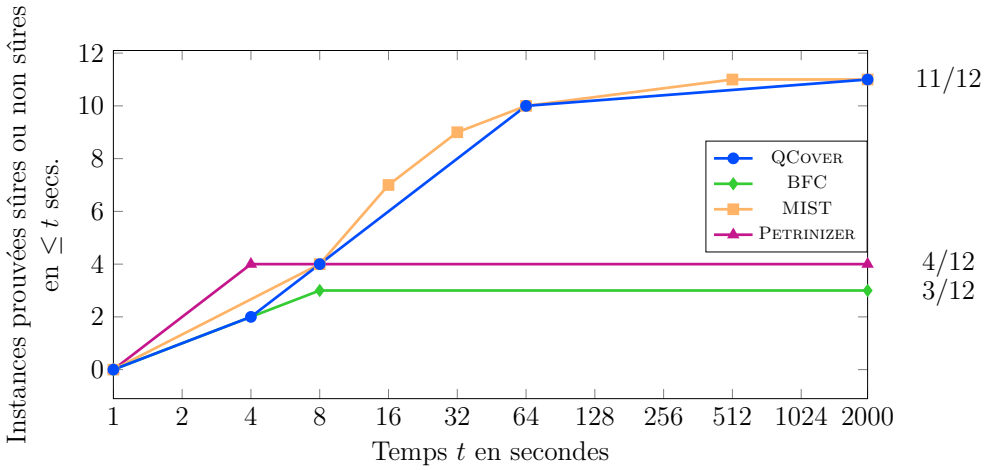


Figure B.5 – Nombre cumulatif d'instances prouvées sûres ou non sûres dans la catégorie `medical`.

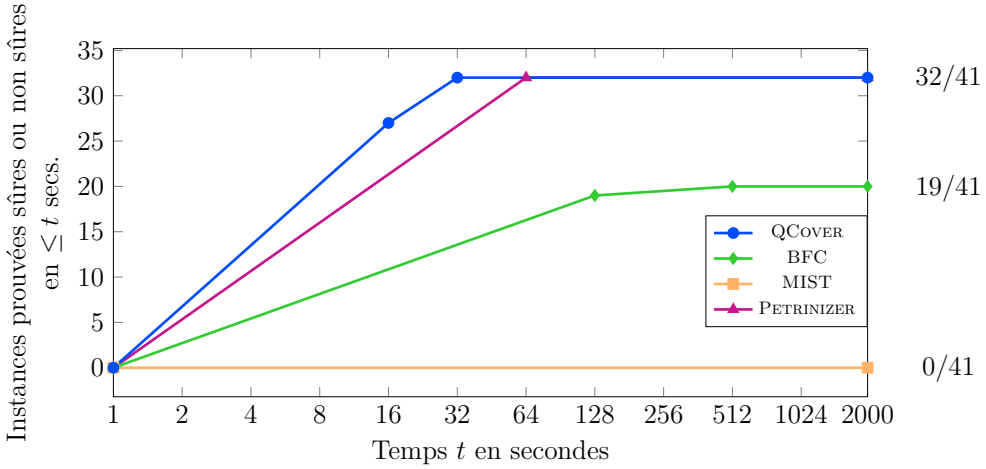


Figure B.6 – Nombre cumulatif d'instances prouvées sûres ou non sûres dans la catégorie `bug_tracking`.

B.4 Comparaison des temps d'exécution sur les instances sûres

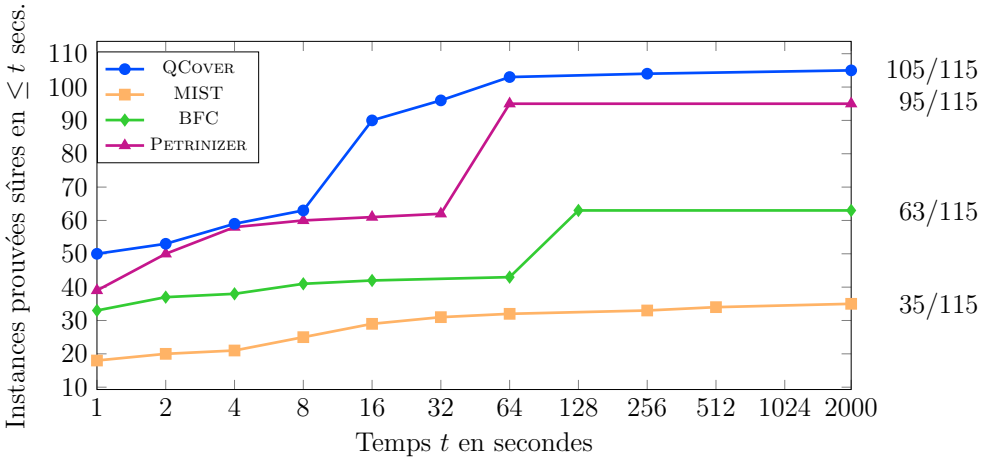


Figure B.7 – Nombre cumulatif d'instances prouvées sûres pour l'ensemble des catégories.

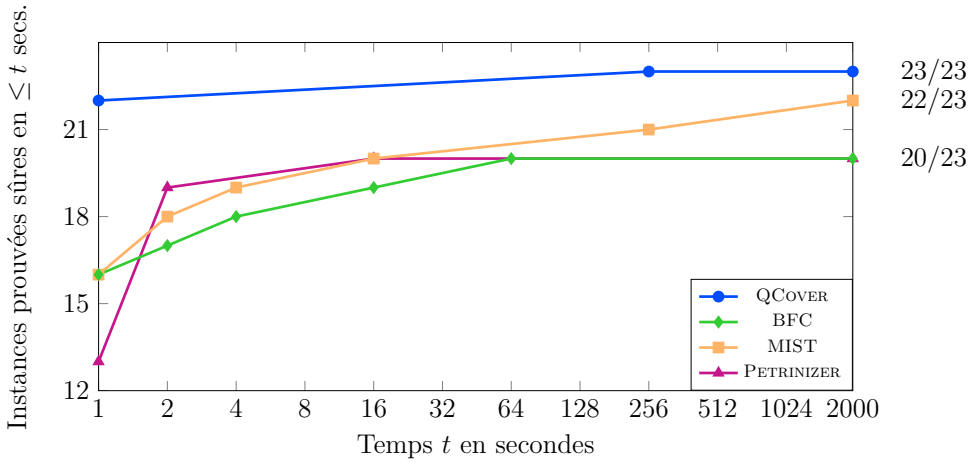


Figure B.8 – Nombre cumulatif d'instances prouvées sûres dans la catégorie `mist`.

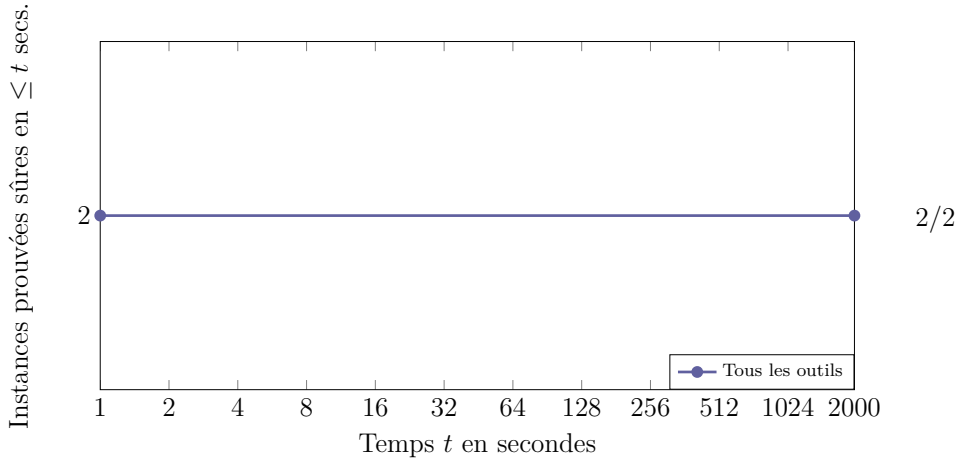


Figure B.9 – Nombre cumulatif d'instances prouvées sûres dans la catégorie `bfc`.

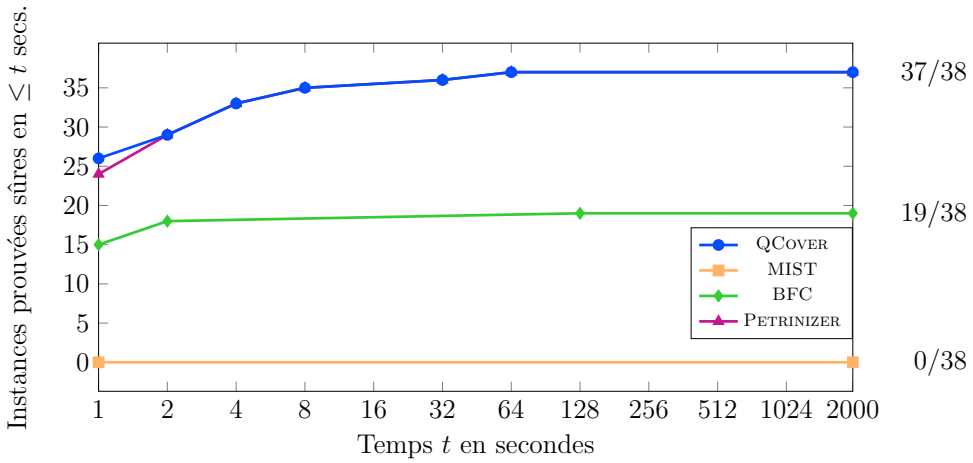


Figure B.10 – Nombre cumulatif d'instances prouvées sûres dans la catégorie `soter`.

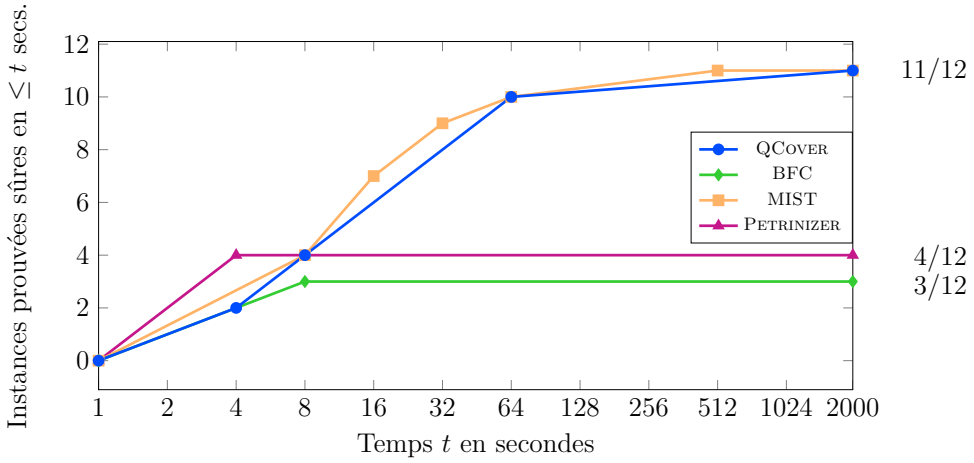


Figure B.11 – Nombre cumulatif d'instances prouvées sûres dans la catégorie `medical`.

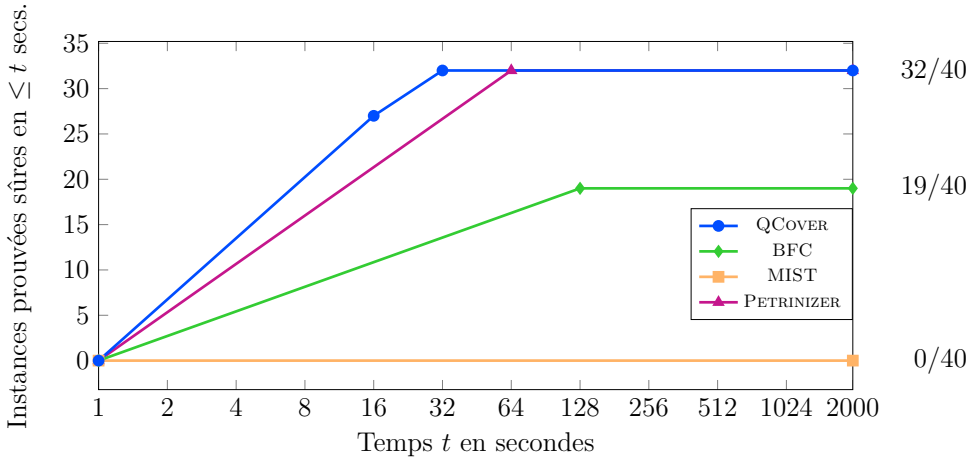


Figure B.12 – Nombre cumulatif d'instances prouvées sûres dans la catégorie `bug_tracking`.

B.5 Comparaison des temps d'exécution sur les instances non sûres

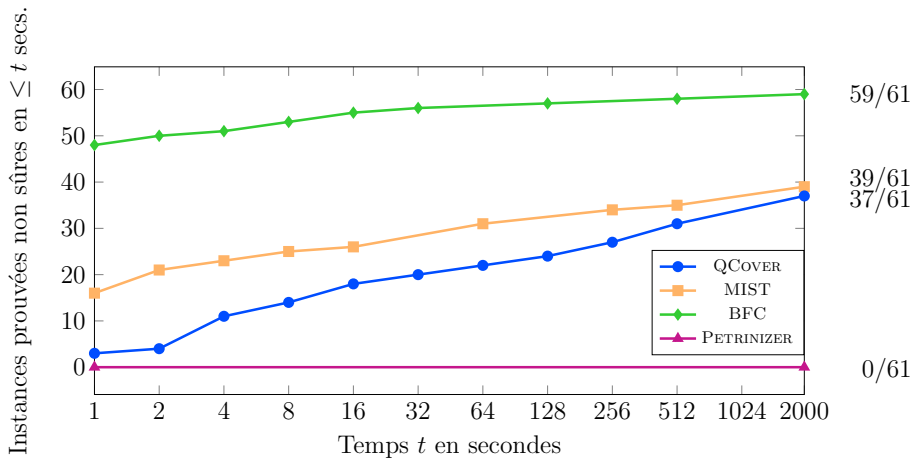


Figure B.13 – Nombre cumulatif d'instances prouvées non sûres pour l'ensemble des catégories.

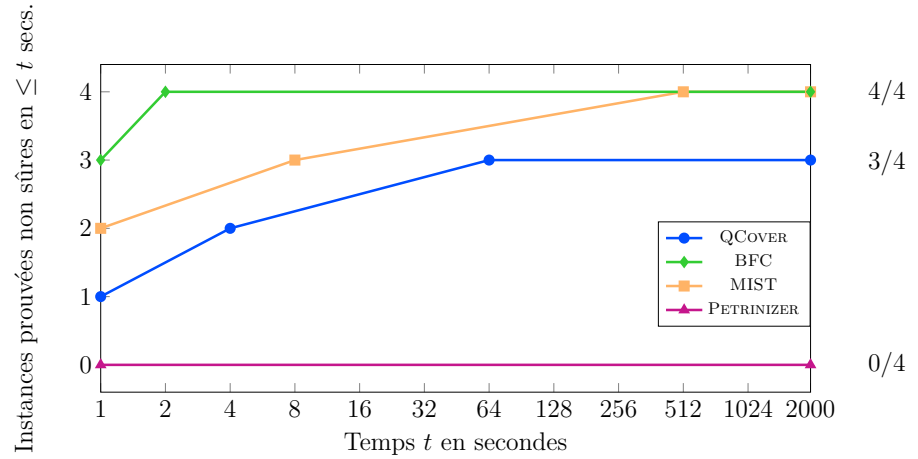


Figure B.14 – Nombre cumulatif d'instances prouvées non sûres dans la catégorie `mist`.

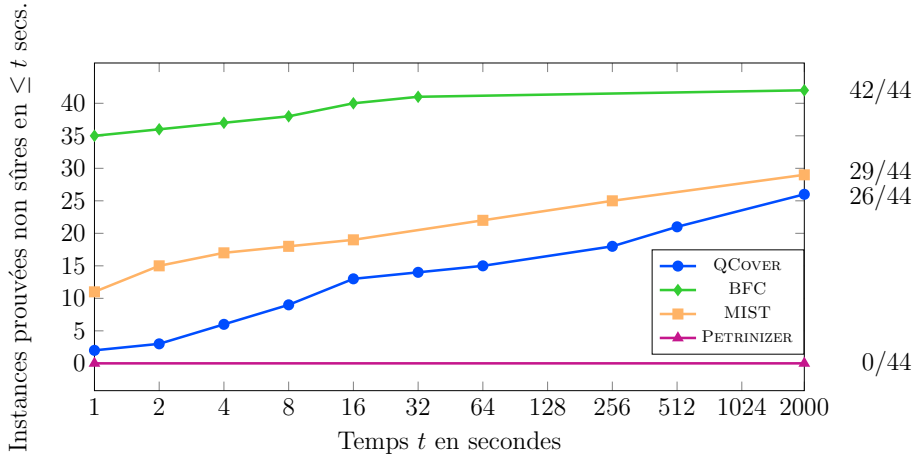


Figure B.15 – Nombre cumulatif d'instances prouvées non sûres dans la catégorie `bfc`.

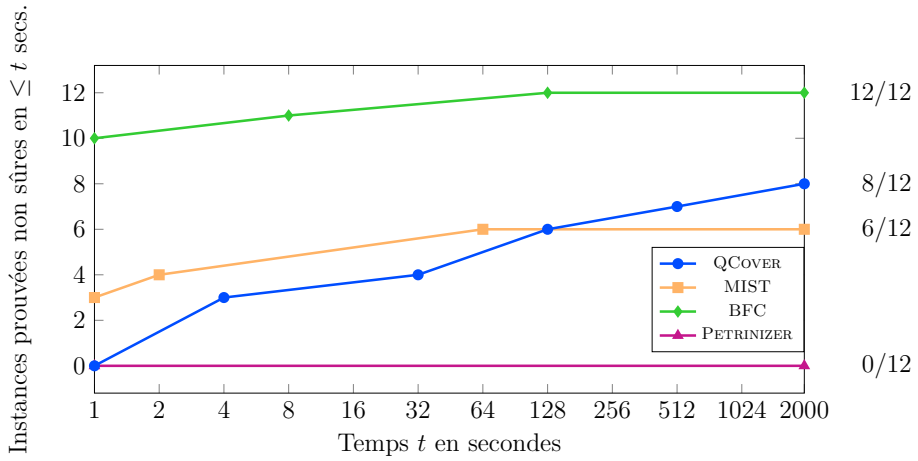


Figure B.16 – Nombre cumulatif d'instances prouvées non sûres dans la catégorie `soter`.

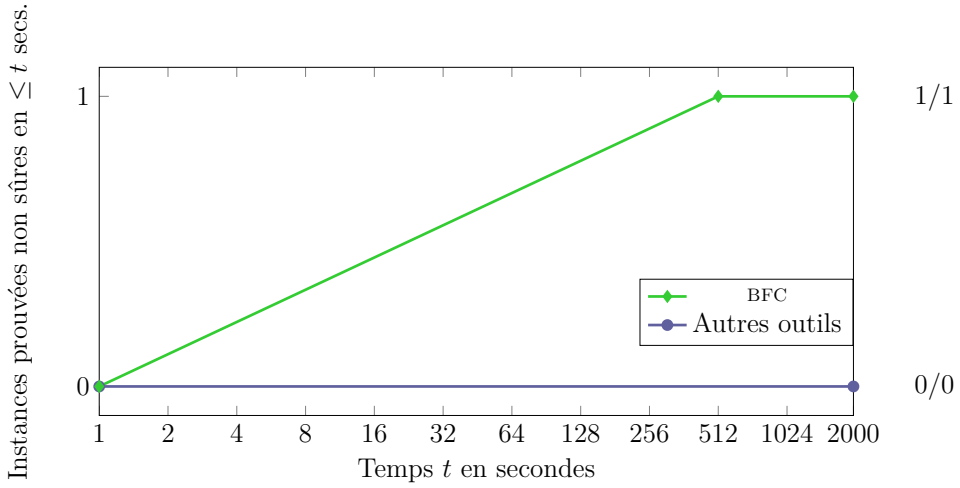


Figure B.17 – Nombre cumulatif d'instances prouvées non sûres dans la catégorie `bug_tracking`.

B.6 Efficacité de l'élimination de nouveaux marquages

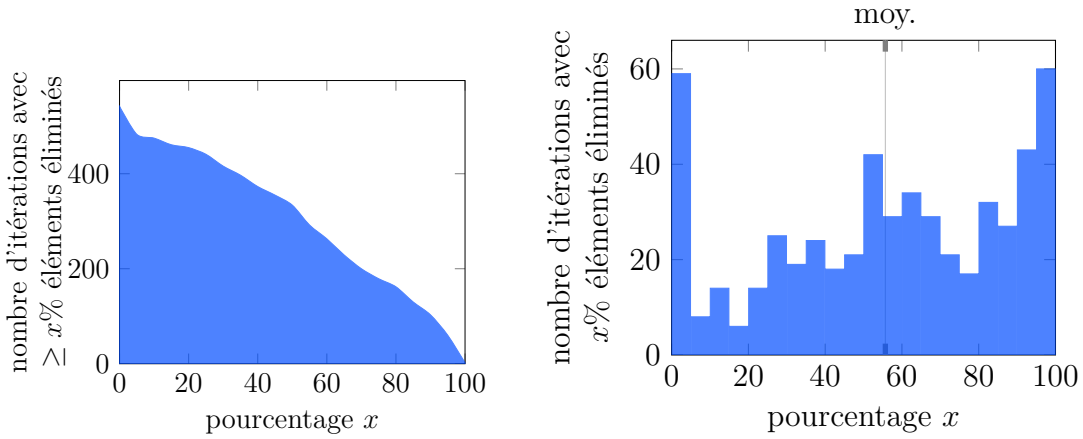


Figure B.18 – Nombre de fois où un certain pourcentage de marquages sont éliminés grâce à l'accessibilité continue.