Contribution à l'étude des jeux sur des graphes de processus à pile

Olivier Serre

Université Paris 7 – Denis Diderot
UFR d'informatique

THÈSE

pour l'obtention du titre de
Docteur de l'Université Paris 7
Spécialité Informatique

présentée par
Olivier SERRE

sur le sujet

Contribution à l'étude des jeux sur des graphes de processus à pile

soutenue publiquement le 30 novembre 2004 devant le jury suivant:

Didier CAUCAL	Rapporteur
Wolfgang THOMAS	Rapporteur
Igor WALUKIEWICZ	Examinateur
Anca MUSCHOLL	Directrice de thèse
Jean-Éric PIN	Directeur de thèse

A Hubert

A Solveig

Remerciements

Il existe de nombreux jeux, mais ils ont tous je crois un point commun: il faut maîtriser certaines connaissances et techniques pour y jouer. Ainsi, parle-t-on d'initiateur, de maître ou encore de mentor. Anca Muscholl et Jean-Éric Pin ont su jouer ce rôle admirablement avec moi et je leur en suis extrêmement reconnaissant. Ils ont su me transmettre leur démon pour ce jeu — sérieux — qu'est la recherche.

J'ai rencontré Anca en maîtrise lors d'un cours inoubliable de complexité. Sa passion m'avait déjà frappé à l'époque. J'ai connu Jean-Éric par ses écrits au cours d'un mémoire sur les langages localement testables. C'est alors tout naturellement que l'année suivante je leur ai demandé de m'encadrer pour mon stage de DEA. Je ne l'ai pas regretté. Je veux ici les remercier pour avoir toujours su m'aider, me soutenir, m'orienter, me conseiller et surtout pour avoir su me laisser une grande liberté et beaucoup d'autonomie tout en gardant un œil bienveillant sur moi.

Je voudrais également remercier Didier Caucal d'avoir accepté de rapporter cette thèse. Il a joué son rôle parfaitement, et ses remarques et corrections m'ont été précieuses.

En décembre 2002, Wolfgang Thomas m'a invité à passer une semaine à Aachen. J'y ai découvert une équipe enthousiaste et à sa tête quelqu'un d'une grande gentillesse. Je veux le remercier d'avoir accepté de rapporter cette thèse malgré un calendrier très chargé. Cette thèse lui doit d'autant plus qu'il est l'un des premiers à avoir vu l'intérêt des jeux sur les graphes infinis.

Igor Walukiewicz m'a fait l'honneur de participer à ce jury. Je veux le remercier pour cela mais aussi pour beaucoup d'autres choses. Tout d'abord pour son article de 1996 sur les jeux sur des graphes de processus à pile qui a occupé une bonne partie de ma première année de thèse. Je voudrais aussi le remercier pour sa gentillesse et pour l'intérêt qu'il a porté à mes travaux depuis trois ans. J'ai eu enfin la chance de travailler avec lui et j'espère que nous aurons par la suite d'autres collaborations.

Outre Igor, je voudrais remercier mes co-auteurs Alexis-Julien Bouquet, Christof Löding et Madhusudan. Travailler avec eux a été stimulant (parfois un peu trop) et le jeu en a toujours valu la chandelle.

Jacques Duparc a joué un rôle particulier dans mes travaux sur les conditions boréliennes. Il a toujours su trouver le temps pour répondre à mes (nombreuses) questions et m'a fait découvrir un domaine fascinant que je ne connaissais pas.

Je voudrais également remercier Damian Niwiński pour l'intérêt qu'il a porté à mes travaux et surtout pour sa gentillesse.

J'ai également une pensée pour tous ceux et celles qui m'ont au cours des années transmis leur passion de l'informatique et des mathématiques: Agnès Bonnier qui m'a montré mon premier automate, Claudine Picaronny qui a encadré mon mémoire de première année à l'ENS Cachan, Alain Finkel, Béatrice Bérard, Antoine Petit, Christian Choffrut, Christiane Frougny et Jacques Sakarovitch. J'ai eu la chance de faire ma thèse dans de bonnes conditions grâce à l'ENS Cachan tout d'abord puis en étant allocataire moniteur ensuite. Je voudrais ainsi remercier Hubert Comon-Lundh et à travers lui le département d'informatique de l'ENS Cachan.

J'ai découvert au LIAFA un lieu de travail particulièrement agréable et stimulant. Je voudrais remercier tout particulièrement Marc Zeitoun et Wieslaw Zielonka pour l'intérêt qu'ils ont porté à mes travaux, Dominique Rossin pour sa ponctualité autour de midi, Noëlle Delgado pour son efficacité, et aussi Laifa Ahmadi. Enfin, comment parler du LIAFA sans avoir une pensés pour les thésards avec qui j'ai partagé (ou non) un bureau durant ces années.

Je voudrais aussi remercier mes parents pour m'avoir fourni depuis très longtemps déjà un cadre familial propice à un épanouissement personnel et scolaire. Il y a près de dix ans maintenant, ils m'ont acheté mon premier ordinateur. Après de nombreuses tentatives, j'avais réussi à les convaincre en leur promettant que j'utiliserai ce dernier non pour jouer mais pour programmer. Je ne sais si cette thèse est la preuve de ma bonne ou de ma mauvaise foi. Je voudrais tout particulièrement remercier mon père qui m'a appris le premier à programmer. Je voudrais aussi remercier ma sœur Caroline, ma grand-mère Simone, mes beaux-parents Claudie et Michel et ma belle sœur Clara.

J'ai aussi une pensée pour Mélanie et Sébastien, les super malins Guillemette et Bart (et leur super raton), Mathilde(s), Mathieu, Gildas et Aïcha, et tout ceux qui à leur manière ont éclairé ces années.

Enfin, tous ceux qui me connaissent un tant soit peu sauront bien que les mots qui suivent sont dérisoires par rapport à ce que j'éprouve pour celle qui m'accompagne depuis toutes ces années. Solveig, je voudrais te remercier ici pour toutes ces heures, celles passées à me relire, celles passées à m'écouter te réciter des exposés en anglais, celles passées à me supporter, celle passées à mes côtés... Merci pour tout ce que tu es et bien plus encore.

Table des matières

Introduction

Les trois premiers chapitres constituent en réalité la véritable introduction de cette thèse. Certes, il ne faut pas cinquantes pages pour présenter les problématiques auxquelles nous allons nous intéresser, mais il aurait été dommage d'écrire une phrase telle que «*nous montrons qu'il existe des conditions de gain de complexité borélienne arbitraire finie qui sont décidables pour n'importe quel jeu sur un graphe de processus à pile*» sans en avoir défini au préalable la plupart des mots. Il ne faut pas voir là une rigueur poussée à l'extrême, mais seulement un souci de clarté. Cette *introduction* se propose donc de définir de façon très générale les problématiques de la vérification en soulignant le rôle que l'étude des jeux à deux joueurs peut y tenir. Les contributions de la thèse sont données dans le chapitre 3 page 65.

La vérification

L'informatique et les systèmes automatisés occupent autour de nous une place chaque jour plus grande. Il en découle que la fiabilité de ces systèmes est un enjeu capital, aussi bien du point de vue économique que du point de vue de la sécurité de chacun d'entre nous. Nous sommes tous conscients de l'importance prise par l'informatique dans notre société. Même si le fameux *bug de l'an 2000* n'a pas été aussi terrible qu'on nous le prédisait, il faut bien reconnaître qu'il n'est pas difficile de dresser une liste de défaillances informatiques, parfois stupides mais difficiles à contourner, rencontrées au cours des dernières années. Aussi la vérification des systèmes revêt-elle un caractère crucial. La conception d'un système passe par de nombreuses étapes qui impliquent une grande diversité d'intervenants. Certains d'entre eux ont une vision assez abstraite du système, tandis que d'autres se confrontent au difficile problème de l'implantation. La conception abstraite du système doit donc être réalisable dans la pratique de façon performante, sous peine de perdre fortement de son intérêt.

Il en va de même pour la vérification. Afin de vérifier un système, on commence par le modéliser. On spécifie ensuite une propriété que l'on souhaite vérifier, dans un formalisme adapté, qui passe le plus souvent par une formule logique. On regarde enfin si le modèle vérifie la spécification. Le problème est double : les phases de modélisation et de spécification doivent permettre une phase de vérification, réalisable non seulement d'un point de vue théorique, mais également d'un point de vue pratique.

La phase de modélisation se doit d'être générale pour pouvoir être réutilisable : il serait dommage de définir un modèle pour un programme faisant trois appels récursifs au plus, et de devoir refaire tout le travail lorsque l'on s'intéresse à un même programme faisant cette fois-ci quatre appels récursifs au plus. Cependant, un modèle trop puissant rendrait la phase de vérification délicate, voire impossible. Une autre notion importante dans la modélisation d'un système est son caractère fini ou non. Par exemple, on peut modéliser un ascenseur dans un immeuble à trois étages par un nombre fini de paramètres, qui peuvent chacun prendre un nombre fini de valeurs. Considérons à présent un programme qui effectue un nombre arbitraire d'appels récursifs : il ne peut être modélisé de façon finie, car la taille de sa pile d'appels récursifs n'est pas bornée. Dans ce cas, on cherchera un modèle finiment descriptible de systèmes infinis.

Quant à la spécification, les outils, logiques ou autres, dépendent fortement du type de propriétés que l'on souhaite vérifier. A nouveau, il serait inadapté d'utiliser un langage de spécification trop puissant qui rendrait la vérification délicate, coûteuse, ou pire irréalisable.

Enfin, la phase de vérification, ou *model-checking*, consiste à décider pour tout modèle d'une forme donnée, et pour toute formule dans une logique donnée, si le modèle satisfait la formule. Il se peut tout d'abord que sous cette forme très générale le problème ne soit pas décidable, c'est-à-dire qu'il n'existe pas d'algorithme permettant d'y répondre. C'est par exemple le cas lorsqu'une logique trop expressive est utilisée, ou lorsque les modèles sont trop puissants. Par ailleurs, il y a souvent un fossé entre une réponse sur le papier et une implémentation réelle de la solution : cette dernière peut s'avérer totalement inutilisable si elle conduit à des temps de réponses beaucoup trop long.

Et les jeux dans tout ça?

On distingue fréquemment deux types de systèmes : les systèmes fermés et les systèmes ouverts. Dans le premier cas, le système n'a pas d'interaction avec l'environnement, alors que dans le second cas l'environnement est actif. Un système ouvert peut représenter un programme et son environnement. Reprenons notre exemple d'ascenseur : le programme est l'ascenseur, tandis que l'environnement est constitué d'utilisateurs qui peuvent appeler l'ascenseur et spécifier un étage où aller. On peut alors se demander si l'ascenseur amène toujours ses passagers à bon port, c'est-à-dire si lorsqu'un étage est demandé, l'ascenseur s'y arrêtera quoi qu'il se passe entre temps (appels de l'ascenseur, plusieurs arrêts demandés,...).

La vérification des systèmes ouverts peut être réalisée à l'aide d'un jeu. On peut représenter le système par un ensembles de sommets reliés entre eux par des transitions. Le modèle du système possède deux types d'états appelés sommets : ceux où la prochaine action sera effectuée par le programme, et ceux où elle sera réalisée par l'environnement. On décide que les sommets où le programme va effectuer la prochaine action appartiennent à un premier joueur, Eve, tandis que les sommets régis par l'environnement appartiennent à un second joueur, Adam. Une partie dans un jeu débute dans le sommet qui correspond à l'état initial du système. Le joueur à qui ce sommet appartient choisit une action à effectuer, et va dans le sommet correspon-

dant à l'état résultant de cette action. Le joueur à qui le nouveau sommet appartient choisit une action et ainsi de suite. Une partie est la simulation d'une exécution du système. Si ce dernier ne se bloque pas, la partie est infinie. Les propriétés que l'on souhaite le plus souvent vérifier sont de la forme «*quoi que fasse l'environnement, l'exécution est une bonne exécution*». On dira alors qu'Eve remporte une partie si celle-ci correspond à une bonne exécution. Sinon c'est Adam qui gagne. Le système vérifie la propriété si toutes les parties sont remportées par Eve.

Une variante du problème de vérification est celui de la synthèse d'un contrôleur. On a cette fois-ci un système ouvert, dans lequel les comportements du système dans un état sont multiples. Par ailleurs, on possède une spécification que l'on veut voir satisfaite, quelle que soit le comportement de l'environnement. En général, les comportements du programme sont trop peu restreints pour cette spécification face à un environnement malicieux. On cherche alors à restreindre les comportements afin que le nouveau système satisfasse la propriété. On parle alors de *synthèse de contrôleur*. En terme de jeu, le problème de synthèse d'un contrôleur correspond à la recherche d'une stratégie gagnante pour Eve. Une stratégie est une manière de jouer, c'est-à-dire une procédure qui décide quelle action choisir, en fonction du sommet courant et du passé de la partie. Une stratégie est gagnante si elle assure la victoire dès lors qu'on la respecte. Si Eve possède une stratégie, on en déduit un contrôleur et réciproquement. Lorsque l'on synthétise un contrôleur, on voudrait, en vue d'une implémentation, que celui-ci soit le plus simple possible. Dans le formalisme des jeux, cela revient à restreindre la mémoire qu'Eve peut utiliser pour calculer sa stratégie. Une autre question importante est de restreindre au minimum le programme. Dans le formalisme des jeux, cela revient à chercher une stratégie la plus permissive possible.

Problématiques de la thèse

Cette thèse s'intéresse davantage à l'étude des jeux, qu'au model-checking. La grande majorité des jeux considérés ici se déroulent sur le même modèle de graphes, celui des graphes de processus à pile et de leurs sous-classes.

L'article [83, 84] d'I. Walukiewicz a été mon point de départ. Le comprendre dans le détail n'a pas été chose aisée, et a demandé de récrire une preuve du résultat principal sur laquelle nombre de constructions présentées dans ce travail se sont fondées.

Les axes de recherches ont été multiples. Le premier a consisté à maîtriser les résultats d'I. Walukiewicz. La question sur laquelle j'ai travaillé par la suite a été celle de la représentation des ensembles de positions gagnantes. L'étude de nouvelles conditions de gain permettant d'exprimer des propriétés naturelles des systèmes a indéniablement été l'un des fils conducteurs de cette thèse. Je me suis également intéressé à une question plus théorique, à savoir l'existence de conditions de gain arbitrairement complexes d'un point de vue topologique, et qui restent décidables. L'étude de modèles de graphes plus simples a enfin retenu mon attention.

Certain des travaux présentés ici ont été le fruit de collaborations fructueuses, avec A.J. Bouquet et I. Walukiewicz d'une part, et avec C. Löding et P. Madhusudan d'autre part. Les conditions boréliennes introduites au sixième chapitre doivent beaucoup à J. Duparc, qui a accepté le rôle d'oracle par courrier électronique in-

terposé. Enfin, cette recherche s'inscrit dans le cadre du projet européen *Games and Automata for Synthesis and Validation* (GAMES), qui lui a offert un contexte stimulant et confortable.

Comment lire cette thèse?

Les trois premiers chapitres sont à considérer comme une longue partie préliminaire à cette recherche. Leur importance et leur intérêt varieront fortement selon les connaissances du lecteur. Le premier chapitre énonce des définitions classiques sur les structures de base que sont les mots, les arbres et les graphes. Divers types d'automates et de langages y sont introduits, tout comme quelques notions de complexité. Pour le lecteur averti, ce chapitre présente l'intérêt de préciser les notations et d'introduire quelques concepts peu courants. Pour le lecteur peu au fait de ces concepts, il constituera, je l'espère, le nécessaire pour lire et comprendre cette thèse. Le deuxième chapitre est le pendant du premier quant aux notions relatives aux jeux. Y sont introduites des notions classiques pour les jeux à deux joueurs, ainsi que des notions plus spécifiques pour les jeux sur des graphes de processus à pile. Toutes les conditions de gain étudiées par la suite y sont présentées. Les notions de complexité borélienne, et de jeu de Wadge y sont abordées. Un lecteur bien au courant de ces notions pourra le parcourir rapidement et s'y reporter simplement lorsque la nécessité le demandera. Pour un lecteur un peu moins familier du domaine, ce chapitre sera, je l'espère, une bonne invitation à lire la suite. Le troisième chapitre dresse enfin un panorama des travaux qui concernent les jeux sur des graphes de processus à pile. Les contributions que ce travail prétend apporter y sont évoquées, tout comme leur place au sein de la thèse.

Les cinq chapitres suivants constituent en revanche le corps de cette thèse. Alors que le quatrième chapitre introduit des concepts qui s'avèrent d'une grande utilité dans tous les autres chapitres, les quatre derniers chapitres sont très largement indépendants. Le quatrième chapitre se penche ainsi plus précisément sur la représentation des ensembles de positions gagnantes. Les résultats qui y sont présentés permettent de simplifier les constructions du reste de la thèse. Le cinquième chapitre s'intéresse quant à lui dans un premier temps à la condition de parité : les résultats d'I. Walukiewicz énoncés dans [83] y sont rappelés. Une nouvelle preuve de la décidabilité de ces jeux est ensuite proposée, avant de s'intéresser à la construction et à la preuve pour les conditions d'explosion et de parité en escalier, qui utilisent les mêmes techniques. Le sixième chapitre traite d'une famille de conditions de gain de complexité borélienne arbitraire finie, pour lesquelles on peut décider le gagnant, tandis que le septième chapitre concerne l'étude des conditions de parité pour des jeux sur des sous-classes des graphes de processus à pile que sont les graphes de BPA et les graphes de processus à compteur. Enfin, le dernier chapitre élargit la recherche à d'autres conditions de gain. L'intérêt se porte dans un premier temps sur les combinaisons booléennes d'une condition de parité et d'une condition sur la hauteur de pile, puis sur les jeux munis d'une condition de gain ω-VPL.

Chapitre 1

Définitions et notations

Sommaire

Dans ce premier chapitre on donne les notions élémentaires qui seront utilisées par la suite. On commence par les objets de base que sont les mots, les arbres et les graphes. On s'intéresse ensuite à la notion de langage de mots puis à celles d'automate fini et d'automate à pile, ce qui nous permet de définir les langages rationnels et les langages algébriques de mots finis et de mots infinis.

La notion d'automate à pile nous permet ensuite d'introduire celle de graphe de processus à pile. On donne également des sous-classes de ces derniers issues des cas particuliers de processus à pile que sont les processus à compteur et les Bpa.

Enfin, on donne quelques notions de complexité et de calculabilité. En particulier, on introduit les différents modèles de machines de Turing et les classes de complexité qui seront utilisées par la suite.

La plupart des notions introduites dans ce chapitre sont classiques. Pour plus de détails, on pourra consulter [44, 70, 73].

1.1 Structures de base

1.1.1 Mots et langages

Mots

Un *alphabet* est un ensemble fini ou infini de symboles. Les éléments d'un alphabet sont des *lettres*. Dans la suite, un alphabet sera le plus souvent désigné par une lettre majuscule romaine, par exemple A,B ou C, et les lettres par des lettres minuscules romaines, par exemple a, b ou c.

Un *mot* sur un alphabet A est une suite de lettres de A. Un mot fini est une suite finie de lettres tandis qu'un mot infini est une suite indexée par les entiers naturels. On représente un mot fini par la juxtaposition des lettres qui le composent. Par exemple $jeux$ représente le mot à quatre lettres associé à la suite j,e,u,x. Le mot vide, c'est-à-dire le mot associé à la suite vide est noté ε.

Etant donnés un mot fini u et un mot (fini ou infini) v, la concaténation de u et de v sera notée $u \cdot v$ ou plus simplement uv, s'il n'y a pas d'ambiguïté.

Un mot fini u est un *préfixe* d'un mot (fini ou infini) v, ce que l'on note $u \sqsubseteq v$, s'il existe un mot w tel que $v = uw$. Un mot fini u est un *suffixe* d'un mot fini v, s'il existe un mot w tel que $v = wu$. Un mot fini u est un *facteur* d'un mot (fini ou infini) v, s'il existe deux mots w_1 et w_2 tels que $v = w_1uw_2$. Enfin, un mot u est un *sous-mot* d'un mot v si la suite des lettres de u est une sous-suite de la suite des lettres de v. Soit un mot u préfixe (respectivement suffixe, facteur ou sous-mot) d'un mot v, u est un préfixe (respectivement suffixe, facteur ou sous-mot) *strict* de v si et seulement si u et v sont distincts. Pour tout mot u de longueur n et tout entier $0 \leq k \leq n$, on désigne par $u\restriction_k$ le préfixe de u de longueur k.

Par exemple, si l'on considère le mot *automate*, *auto* en est un préfixe, *tomate* un suffixe, *ma* un facteur et *atome* un sous-mot.

On note $|u|$ la longueur (éventuellement infinie) du mot u. Si a est une lettre, $|u|_a$ désigne le nombre d'occurrences de a dans u.

Etant donnés un mot u et un entier $n \geq 0$, on désigne par u^n le mot formé de n concaténations de u. En particulier u^0 est le mot vide ε.

Etant donné un mot fini $u = a_1a_2 \cdots a_n$, on appelle *miroir* de u, le mot $\widetilde{u} = a_na_{n-1} \cdots a_1$. Par exemple, si $u = olivier$, $\widetilde{u} = reivilo$. Un mot égal à son miroir sera appelé *palindrome*. Les mots *anna* et *esoperesteicietserepose* sont des exemples de palindromes.

Soit un alphabet A. On note A^* l'ensemble des mots finis sur A, A^+ l'ensemble des mots finis différents du mot vide, A^ω l'ensemble des mots infinis sur A, et A^∞ l'ensemble des mots finis ou infinis sur A.

Enfin, étant donnée une suite $(u_i)_{i\geq 1} \in (A^*)^{\mathbb{N}}$ de mots finis sur un alphabet A, on définit le mot, fini ou infini, $\lim(u_i)_{i\geq 1}$, appelé *limite de la suite* $(u_i)_{i\geq 1}$, et défini comme le mot maximal pour l'inclusion vérifiant ce qui suit : pour tout $j \geq 1$, il existe un entier r telle que la j-ème lettre de $\lim(u_i)_{i\geq 1}$ est la même que la j-ème lettre de u_p pour tout $p \geq r$.

Langages

Un *langage* désigne un ensemble de mots et sera, en général, noté par une lettre majuscule romaine, par exemple L.

On distinguera deux cas particuliers de langages : les langages de mots finis et les langages de mots infinis, également appelés *ω-langages* (la lettre ω faisant référence au caractère infini des mots de ces langages). Un langage de mots finis sur un alphabet A sera donc un sous-ensemble de A^*, tandis qu'un ω-langage sur l'alphabet A sera un sous-ensemble de A^ω. La grande majorité des langages considérés par la suite seront de l'une des deux formes précédentes.

Un langage L est *fermé par préfixe* lorsque pour tout mot u dans L, tout préfixe de u est dans L.

Plusieurs opérations peuvent être effectuées sur les langages. Les langages étant des ensembles, on rappelle les opérations booléennes :

- étant donnés deux langages L_1 et L_2, le langage $L = L_1 \cup L_2 = \{u \in A^\infty \mid u \in L_1 \text{ ou } u \in L_2\}$ est appelé *union* de L_1 et L_2.
- étant donnés deux langages L_1 et L_2, le langage $L = L_1 \cap L_2 = \{u \in A^\infty \mid u \in L_1 \text{ et } u \in L_2\}$ est appelé *intersection* de L_1 et L_2.
- étant donnés deux langages L_1 et L_2, le langage $L = L_1 \setminus L_2 = \{u \in A^\infty \mid u \in L_1 \text{ et } u \notin L_2\}$ est appelé *différence* de L_1 et L_2.
- étant donné un langages L, le langage $\overline{L} = A^\infty \setminus L$ est appelé *complémentaire* de L. Lorsque L est un langage de mots finis (respectivement de mot infinis) et que le contexte est clair, on appellera également complémentaire le langage $A^* \setminus L$ (respectivement $A^\omega \setminus L$).

Outre les opérations booléennes, il existe d'autres opérations naturelles sur les langages, fondées sur la notion de concaténation pour les mots :

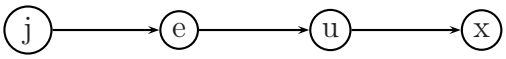
- étant donnés un langage L_1 de mot finis et un langage L_2 quelconque, le langage $L = L_1L_2$ est le langage $\{u_1u_2 \in A^\infty \mid u_1 \in L_1 \text{ et } u_2 \in L_2\}$. Le langage L est appelé *produit* de L_1 et L_2.
- étant donné un langage L de mots finis, on définit par récurrence la suite de langages finis $(L^i)_{i\geq 0}$ par $L_0 = \{\varepsilon\}$ et $L^{i+1} = L^iL$. On parle alors de *puissances d'un langage.*
- étant donné un langage de mots finis L, l'union des puissances de L, notée L^* et définie par $L^* = \bigcup_{i\geq 0} L^i$, est appelée *étoile de L*. Une variante consiste à ne considérer que les puissances non nulles de L, c'est-à-dire à poser $L^+ = \bigcup_{i>0} L^i$.
- étant donné un langage de mots finis L ne contenant pas le mot vide, le langage de mots infinis $L^\omega = \{u = u_0u_1u_2\cdots \mid (u_i)_{i\geq 0} \in L^{\mathbb{N}}\}$ est appelé *ω-itération de L*.
- étant donné un langage de mots finis L ne contenant pas le mot vide, le langage $L^\infty = L^* \cup L^\omega$ est appelé *∞-itération de L*.

Etant donné un alphabet A, les notations A^*, A^ω et A^∞ prennent tout leur sens au regard des définitions précédentes en considérant A comme un langage de mots réduits à une lettre.

1.1.2 Arbres

Nous avons défini un mot comme une suite de lettres. Ainsi, un mot fini de longueur n sur un alphabet A peut-il être vu comme une application de $\{0,\ldots,n-1\}$ dans A, associant à tout i, $0 \leq i \leq n-1$, la $(i+1)$-ème lettre du mot. De même, un mot infini sur un alphabet A peut être considéré comme une application de $\mathbb{N}$ dans A. La structure naturelle de $\mathbb{N}$, munie de la relation successeur, confère alors aux mots un caractère linéaire.

Les listes chaînées fournissent un représentation naturelle des mots : la première lettre du mot est distinguée et chaque lettre pointe vers la lettre suivante. Ainsi, le mot *jeux* peut-il être représenté par la liste chaînée suivante :



Dans cette optique, une généralisation naturelle des mots consiste à considérer qu'il y a une unique lettre de départ, qu'une lettre ne pointe, non plus vers une seule lettre, mais vers plusieurs lettres ordonnées entre elles et que deux lettres ne peuvent pointer sur une même troisième. Ainsi, la structure n'est plus linéaire mais arborescente. Graphiquement, on obtient par exemple la figure 1.1.

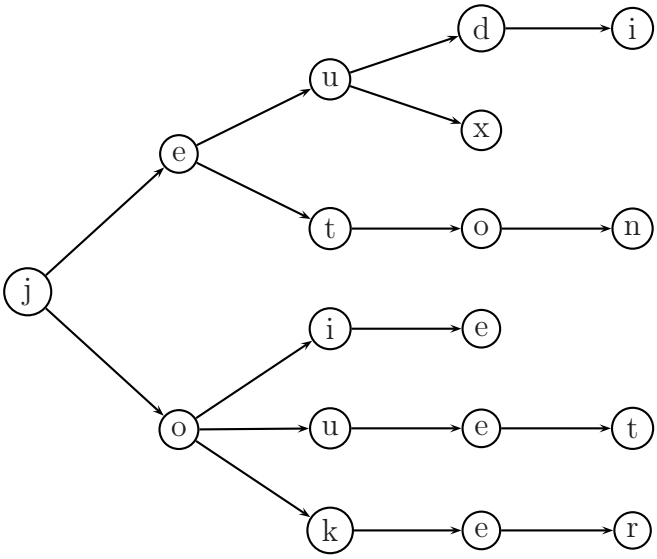


FIG. 1.1 – *Un exemple d'arbre*

Alors que dans le cas des mots on indexait les lettres par un ensemble totalement ordonné ($\{0,\ldots,n-1\}$ pour les mots finis de longueur n et $\mathbb{N}$ pour les mots infinis), ce qui permettait de définir au plus un successeur à chaque lettre, dans le cas des arbres, ce n'est plus le cas et une lettre peut avoir plusieurs successeurs. En fait, on indexe maintenant les lettres par les éléments d'un ensemble, muni d'un ordre partiel, possédant un élément minimal et tel que deux éléments incomparables ne peuvent être plus petit qu'un même troisième.

Plus précisément, considérons un alphabet K muni d'un ordre total $<$, et un langage $D \subseteq K^\infty$ fermé par préfixe. L'ensemble D muni de la relation préfixe est

un ensemble partiellement ordonné, que l'on appelle *domaine*. Un *arbre* d'alphabet A et de domaine D est une application $\mathcal{T}$ de D dans A.

Lorsque K est fini, on identifie K avec $\{0,\ldots,k-1\}$ et l'on parle alors d'arbre *d'arité finie* k. Dans le cas contraire, on parle d'arbre *d'arité infinie*. Lorsque $D \subseteq K^*$ avec K fini, on parle *d'arbre fini*, et dans le cas contraire *d'arbre infini*. En particulier, tout arbre fini est d'arité finie.

Une représentation graphique d'un arbre est la suivante : à tout élément $d \in D$ on associe un nœud, dont l'*étiquette* est $\mathcal{T}(d)$. On appelle d l'*indice* du nœud. Enfin, un *arc* (orienté) relie le nœud d'indice d à tous les nœuds d'indice dh, pour $h \in K$, lorsqu'ils existent. Lorsqu'il y a un arc d'un nœud d'indice i vers un nœud d'indice ih, on dira du premier nœud qu'il est le *père* du second et du second qu'il est le *h-fils* du premier (ou simplement le *fils* selon le contexte). La clôture transitive de la relation père/fils définit la relation *ancêtre/descendant*.

Un arbre $\mathcal{T}$ d'alphabet A et de domaine $D \subseteq K^\infty$ est *équivalent* à un arbre $\mathcal{T}'$ d'alphabet A' et de domaine $D' \subseteq K'^\infty$ si et seulement s'il existe un morphisme lettre à lettre φ de K^∞ dans K'^∞ préservant les ordres sur K et K', et tel que $\varphi(D) = D'$, et pour tout $d \in D$, $\mathcal{T}(d) = \mathcal{T}'(\varphi(d))$. Cette relation est bien entendu une relation d'équivalence et dès lors on confondra tous les arbres d'une même classe.

Un nœud qui n'a pas de fils est une *feuille*. L'unique nœud à n'avoir pas de père est le nœud indexé par le mot vide. Il est qualifié de *racine*, et il est l'ancêtre de tous les nœuds. Un *chemin* dans un arbre est une suite de nœuds reliés par des arcs. Enfin, une *branche* est un chemin maximal partant de la racine. Une branche peut être finie ou infinie. On a le résultat suivant.

Lemme 1.1 (Koening) *Tout arbre infini d'arité finie possède une branche infinie.*

Exemple 1.1 Considérons l'arbre fini d'arité 3 représenté par la figure 1.1. En reprenant les notations ci-dessous, on a $K = \{0,1,2\}$ et $A = \{d,e,i,j,k,n,o,r,t,u,x\}$. Avec la convention où l'on dessine le 0-fils en bas, le 1-fils au milieu et le 2-fils en haut par rapport à leur père, le nœud d'indice 0011 a pour étiquette r et est une feuille. Enfin, la racine est étiquetée par j.

Il est facile de voir que les mots sont un cas particulier d'arbre où K est un singleton. Par analogie avec le mot vide, on définit l'*arbre vide* comme étant l'arbre dont le domaine est vide. De même, on donne un analogue de la relation de suffixe dans le cadre des arbres. Soit donc un arbre $\mathcal{T}$ de domaine D et d'alphabet A. Un arbre $\mathcal{T}'$ de domaine D' et d'alphabet A est un *sous-arbre* de $\mathcal{T}$ s'il existe un élément $d \in D$ tel que $\mathcal{T}'$ soit équivalent à l'arbre $\mathcal{T}''$ de domaine $D'' = d^{-1}D = \{d'' \mid dd'' \in D\}$, d'alphabet A et défini par $\mathcal{T}''(d'') = \mathcal{T}(dd'')$.

Graphiquement, l'arbre $\mathcal{T}''$ (et donc $\mathcal{T}'$) correspond à l'arbre obtenu en ne gardant dans la représentation $\mathcal{T}$ que le nœud d'indice d et ses descendants.

1.1.3 Graphes

Nous venons de définir les arbres comme des applications d'un domaine D muni d'un ordre partiel tel qu'il existe un élément minimal et possédant la propriété que deux éléments incomparables ne peuvent être plus petits qu'un même troisième. Si

l'on enlève la contrainte concernant l'existence d'un élément minimal, on obtient une *forêt*, c'est-à-dire une union disjointe d'arbre. Si l'on indexe seulement par un ordre partiel (sans autre contrainte), on obtient un *graphe dirigé sans circuit*. Plus généralement, on définit un graphe en indexant les sommets (qui sont aux graphes ce que les nœuds sont aux arbres) par les éléments d'un ensemble muni d'une relation binaire.

Un *graphe* est donc un couple $G = (V,E)$, où V est un ensemble de *sommets* et $E \subseteq V \times V$ une relation binaire sur V appelée *arcs*.

La définition précédente ne décrit que la structure et pas un étiquetage éventuel des sommets. Cela dit, il n'est pas difficile d'adjoindre à un graphe une application associant à tout sommet une étiquette prise dans un ensemble donné. Ceci permet alors de voir tout arbre comme un cas particulier de graphe. Cependant, dans l'utilisation que nous allons faire des graphes, nous n'utilisons pas d'étiquetage des sommets, et dès lors nous adoptons la convention qui veut que, dans un graphe, les sommets ne soient pas étiquetés.

Enfin, on définit une notion plus générale dans laquelle on étiquette les arcs. Un *graphe L-étiqueté* est un couple $G = (V,E)$ où V est comme précédemment un ensemble de noeuds, L est un ensemble fini d'étiquettes sur les arcs, et $E \subseteq V \times L \times V$ est une relation ternaire définissant les arcs et leurs étiquettes.

On considérera également des *graphes partiellement étiquetés*, c'est-à-dire des couples $G = (V,E)$, où V est comme précédemment un ensemble de sommets, L un ensemble fini d'étiquettes, et $E \subseteq V \times (L \cup \{\varepsilon\}) \times V$ une relation ternaire définissant les arcs et leurs étiquettes. Une étiquette étiquetée par le mot vide ε est considérée comme non étiquetée.

Lorsque le contexte est clair, nous confondrons les notions de graphe, de graphe étiqueté et de graphe partiellement étiqueté. Etant donné un graphe $G = (V,E)$ et un arc $e \in E$, le premier élément de e est appelé *origine* et le dernier est appelé *extrémité*.

Deux graphes non étiquetés $G = (V,E)$ et $G' = (V',E')$ sont équivalents s'il existe une bijection φ de V sur V' telle que pour tout $(v,v') \in V^2$, $(v,v') \in E$ si et seulement si $(\varphi(v),\varphi(v')) \in E'$. Deux graphes L-étiquetés $G = (V,E)$ et $G' = (V',E')$ sont équivalents s'il existe une bijection φ de V dans V' telle que pour tout $(v,v') \in V^2$ et pour tout $l \in L$, $(v,l,v') \in E$ si et seulement si $(\varphi(v),l,\varphi(v')) \in E'$.

Un graphe est dit fini lorsqu'il possède un nombre fini de sommets, dans le cas contraire, il sera qualifié d'infini.

Un *sous-graphe* d'un graphe $G = (V,E)$ est un graphe $G' = (V',E')$ tel que $V' \subseteq V$ et $E' = E \cap (V' \times V')$. Ainsi, un sous-graphe est obtenu en ne gardant qu'un sous-ensemble de sommets, dont on ne modifie pas l'étiquetage, et tous les arcs existant entre ceux-ci.

On adopte fréquemment la représentation suivante d'un graphe : un sommet v est représenté par un cercle dans lequel on écrit v, et un arc (v_1,v_2) par une flèche partant du sommet v_1 et allant au sommet v_2. Dans le cas où G est étiqueté, on note au-dessus des flèches les étiquettes.

Exemple 1.2 La figure 1.2 donne une représentation du graphe $\{a,b\}$-étiqueté $G =$

(V,E) avec :
- $V = \{1,2,3,4\}$.
- $E = \{(1,a,2),(1,b,2),(1,a,4),(2,a,1),(2,b,2),(4,a,2)\}$.

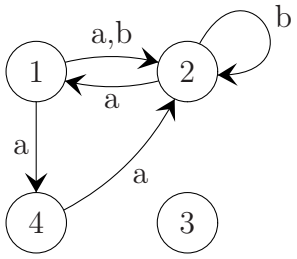


FIG. 1.2 – *Représentation d'un graphe*

Un *chemin* dans un graphe $G = (V,E)$ est un couple (v,c) formé d'un sommet $v \in V$ et d'un mot fini ou infini $c = e_1e_2e_3\cdots$ sur l'alphabet des arcs E tel que v est l'origine de e_1, et tel que pour tout entier $i \geq 1$, l'extrémité de e_i coïncide avec l'origine de e_{i+1}. On appelle v l'*origine* du chemin. Dans le cas d'un chemin fini, l'extrémité du dernier arc est appelé *extrémité* du chemin. Dans le cas des graphes étiquetés, on appellera *étiquette* le mot obtenu en lisant les étiquettes des arcs composant le chemin. La *longueur* d'un chemin (v,c), notée $|(v,c)|$, est la longueur de c. On dit d'un chemin fini, d'origine v et d'extrémité v', qu'il *relie* le sommet v au sommet v'.

Dans le graphe représenté par la figure 1.2, $[1,(1,a,4)(4,a,2)(2,b,2)(2,b,2)(2,a,1)]$ est un chemin de longueur 5 d'origine 1 et d'extrémité 1. Le couple $[4,(4,a,2)(2,b,2)^\omega]$ est un chemin infini dans G d'origine 4.

Un graphe est qualifié de *fortement connexe* s'il existe, pour tout couple de sommets distincts $(v,v') \in V^2$, un chemin de longueur non nulle reliant v à v'. Le graphe précédent n'est pas fortement connexe puisqu'il n'existe pas de chemin reliant 3 à 4.

Un *circuit* dans un graphe $G = (V,E)$ est un chemin fini de longueur non nulle dont l'origine coïncide avec l'extrémité. Dans le graphe représenté par la figure 1.2, $[1,(1,a,4)(4,a,2)(2,a,1)]$ est un circuit. Un graphe est qualifié de *sans circuit* s'il ne possède pas de circuit.

Notation 1.1 *Pour expliciter le caractère relationnel des arcs, un graphe (V,E) sera parfois noté $(V,\rightarrow)$, et le fait que l'on ait $(v,v') \in E$ sera noté $v \rightarrow v'$. Dans le cas des graphes étiquetés , on notera $v \xrightarrow{a} v'$ pour signifier que $(v,a,v') \in E$.*

Remarque 1.1 Dans le cas particulier d'un graphe $G = (V,E)$ non étiqueté et étant donnés deux noeuds, il existe au plus un arc allant de l'un vers l'autre. Appelons ζ l'application de E dans V associant à tout arc son extrémité. L'application ζ définit un morphisme lettre à lettre (également noté ζ) de E^∞ dans V^∞. Ce dernier s'étend en une application ξ de $V \times E^\infty$ dans V^∞ en posant $\xi((v,c)) = v \cdot \zeta(c)$. Il est alors

facile de voir que ξ est une bijection des chemins dans G dans le sous-ensemble de V^∞ défini par :

$$\{c = v_1v_2v_3\cdots \in V^\infty \mid \forall i,\ 1 \leq i < |c|,\ (v_i,v_{i+1}) \in E\}$$

Dès lors, dans le cas des graphes non étiquetés, on peut adopter la définition suivante pour les chemins : un *chemin* est un mot $c = v_1v_2v_3\cdots$ fini ou infini sur l'alphabet V tel que pour tout $1 \leq i < |c|$, $(v_i,v_{i+1}) \in E$.

1.2 Langages et automates

1.2.1 Langages rationnels de mot finis

Comment décrire un langage de mots finis? Les langages étant des ensembles potentiellement infinis et arbitrairement complexes, ils ne peuvent pas être tous descriptibles par un modèle raisonnablement simple. Une solution consiste à construire des langages et leur représentation, de façon récurrente à partir de langages de base et d'opérations que l'on qualifiera *d'opérations rationnelles*. En procédant de cette façon, nous n'obtenons pas tous les langages, mais nous obtenons une classe intéressante, que l'on appelle *langages rationnels*.

Définition 1.1 *Etant donné un alphabet A, la classe des* langages rationnels *sur A est la plus petite classe $Rat\ A^*$ de langages satisfaisant les propriétés suivantes :*

- $\emptyset \in Rat\ A^*$.
- $\{a\} \in Rat\ A^*$ *pour toute lettre* $a \in A$.
- $Rat\ A^*$ *est fermé pour l'union, le produit et l'étoile.*

La définition ci-dessus donne donc une classe de langages construits de façon récurrente à partir de langages très simples et en utilisant des opérations de base. Pour représenter de tels langages, on utilise les *expressions rationnelles*

Définition 1.2 *Les expressions rationnelles et leur langage rationnel associé, sont définis par récurrence :*

- $\emptyset$ *est une expression rationnelle et est associée au langage rationnel* $\emptyset$.
- a *est une expression rationnelle et est associée au langage rationnel* $\{a\}$ *pour toute lettre* $a \in A$.
- ε *est une expression rationnelle et est associée au langage rationnel* $\{\varepsilon\}$.
- *si* E_1 *et* E_2 *sont des expressions rationnelles respectivement associées aux langages* L_1 *et* L_2, $(E_1 \cup E_2)$, $(E_1 \cdot E_2)$ *et* E_1^* *sont des expressions rationnelles respectivement associées aux langages rationnels* $L_1 \cup L_2$, $L_1 \cdot L_2$ *et* L_1^*.

Lorsqu'il n'y a pas d'ambiguïté, on supprime les parenthèses inutiles.

Dans la suite, on se permettra l'abus de langage consistant à confondre une expression rationnelle avec le langage qu'elle représente. Ainsi, on dira par exemple *considérons le langage* $L = (aa)^*$ pour signifier *considérons le langage L représenté par l'expression rationnelle* $(aa)^*$.

Automates finis et langages reconnaissables

La notion d'expression rationnelle, permet de représenter de façon finie un langage rationnel de mots finis. Une autre approche pour représenter un langage rationnel est l'utilisation des automates finis qui peuvent être vus comme des machines mangeant un mot lettre à lettre et décidant, une fois celui-ci ingurgité, s'il appartient ou non au langage représenté par l'automate.

De façon plus formelle, on définit les *automates finis*.

Définition 1.3 *Un automate fini $\mathcal{A}$ est un quintuplet $\langle Q,A,I,F,\delta\rangle$, où Q est un ensemble fini d'états, A est un alphabet fini, $I \subseteq Q$ est un sous-ensemble de Q appelé ensemble des états initiaux, $F \subseteq Q$ est un sous-ensemble de Q appelé ensemble des états finaux, et δ est une application de $Q \times A$ dans les parties $\mathcal{P}(Q)$ de Q appelée fonction de transition.*

Les automates seront représentés en général par une lettre majuscule calligraphiée, par exemple $\mathcal{A}$ ou $\mathcal{B}$.

La représentation graphique d'un automate $\mathcal{A} = \langle Q,A,I,F,\delta\rangle$ est celle du graphe A-étiqueté ayant Q pour ensemble de sommets, et ayant pour arcs l'ensemble des triplets (q,a,q') pour tout $q,q' \in Q$, $a \in A$ tels que $q' \in \delta(q,a)$. Enfin, on ajoute une flèche entrante à chaque sommet associé à un état initial et une flèche sortante à chaque sommet associé à un état final.

Exemple 1.3 Soit $\mathcal{A} = \langle Q,A,I,F,\delta\rangle$ l'automate défini par :

- $Q = \{p,q,r\}$.
- $A = \{a,b\}$.
- $I = \{p\}$.
- $F = \{p\}$.
- $\delta(p,a) = \{q\}$, $\delta(p,b) = \{r\}$, $\delta(q,a) = \{p,q\}$, $\delta(q,b) = \{r\}$, $\delta(r,a) = \{r\}$ et $\delta(r,b) = \{r\}$.

La représentation graphique de cet automate est donnée par la figure 1.3.

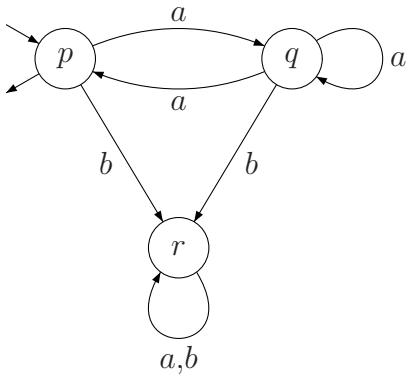


FIG. 1.3 – *Représentation graphique d'un automate*

Définition 1.4 *Etant donné un automate $\mathcal{A} = \langle Q,A,I,F,\delta\rangle$, on dit qu'il y a une transition d'un état p à un état q d'étiquette a, ce que l'on note $p \xrightarrow{a} q$ si $q \in \delta(p,a)$.*

Un calcul $c : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots q_{n-1} \xrightarrow{a_n} q_n$ *est une suite de transitions telles que l'origine de chacune coïncide avec l'extrémité de la précédente. Le mot* $a_1a_2\cdots a_n$ *est appelée* l'étiquette *du calcul c. On dit également que c est un calcul de* q_0 *à* q_n *d'étiquette* $a_1a_2\cdots a_n$*. L'état* q_0 *est appelé* origine *de c, tandis que* q_n *est* l'extrémité *de c. Enfin, un calcul acceptant est un calcul dont l'origine est un état initial et dont l'extrémité est un état final.*

Définition 1.5 *Un mot fini u est* accepté *(ou reconnu) par un automate* $\mathcal{A}$ *s'il est l'étiquette d'au moins un calcul acceptant de* $\mathcal{A}$*. On note* $L(\mathcal{A})$ *l'ensemble des mots acceptés par l'automate* $\mathcal{A}$*, et* $L(\mathcal{A})$ *est appelé* langage reconnu par $\mathcal{A}$*.*

Exemple 1.4 Le langage reconnu par l'automate de la figure 1.3 est décrit par l'expression rationnelle $a(a^*)a$.

Les automates fournissent donc un autre moyen de représenter des langages de mots finis. Un langage reconnu par un automate fini est qualifié de *reconnaissable.*

Définition 1.6 *Soit L un langage de mots finis. On dit que L est un langage* reconnaissable *s'il existe un automate fini qui le reconnaît. Etant donné un alphabet A, on note Rec* A^* *l'ensemble des langages reconnaissables sur l'alphabet A.*

Quel est le lien entre les langages rationnels et les langages reconnaissables sur un alphabet fini A? La réponse nous est donnée par le théorème de Kleene

Théorème 1.1 (Kleene) *Pour tout alphabet fini A, les langages rationnels de A et les langages reconnaissables de A coïncident :*

$$Rat\ A^* = Rec\ A^*.$$

Automates déterministes

Considérons l'automate $\mathcal{A}$ de la figure 1.3. Le mot *aaa* est accepté par $\mathcal{A}$, puisque le calcul $p \xrightarrow{a} q \xrightarrow{a} q \xrightarrow{a} p$ est un calcul acceptant d'étiquette *aaa*. Cependant, le calcul $p \xrightarrow{a} q \xrightarrow{a} p \xrightarrow{a} q$ est un calcul d'étiquette *aaa* qui commence dans un état initial, mais qui n'est pas acceptant.

Ainsi et étant donné un automate, il peut y avoir plusieurs calculs de même étiquette dont certains sont acceptants et d'autres non. Cela provient du fait que δ est une application à valeur dans $\mathcal{P}(Q)$ et que I n'est pas forcément réduit à un singleton. D'un point de vue algorithmique, il serait plus simple de ne pas avoir cette ambiguïté que l'on appelle *non-déterminisme.* Pour cela, on définit les automates *déterministes.*

Définition 1.7 *Soit* $\mathcal{A} = \langle Q,A,I,F,\delta\rangle$ *un automate. L'automate* $\mathcal{A}$ *est un* automate déterministe *s'il vérifie les deux conditions suivantes :*

1. *I est un singleton.*
2. *pour tout état* $q \in Q$ *et toute lettre* $a \in A$, $\delta(q,a)$ *contient au plus un élément.*

En terme de représentation graphique, la seconde condition signifie qu'il ne peut y avoir deux flèches de même étiquette sortant d'un même sommet.

Les automates déterministes reconnaissent les mêmes langages que les automates généraux. Cependant, le prix à payer est qu'ils sont moins concis.

Théorème 1.2 *Soit L un langage reconnu par un automate $\mathcal{A}$. Il existe un automate déterministe $\mathcal{A}_{det}$ qui reconnaît L.*

Pour tout entier n, il existe un langage L, reconnu par un automate non déterministe à n états, et tel que tout automate déterministe reconnaissant L possède au moins 2^n états.

Notation 1.2 *On se permettra, lorsqu'il n'y a pas d'ambiguïté, de noter $\mathcal{A} = \langle Q,A,i,F,\delta\rangle$ au lieu de $\mathcal{A} = \langle Q,A,\{i\},F,\delta\rangle$.*

Automates alternants

Les automates non déterministes peuvent être vus comme une généralisation (qui n'augmente pas le pouvoir de reconnaissance) des automates déterministes. Cette généralisation introduit un caractère existentiel au modèle : un mot est accepté s'il existe un calcul acceptant (alors que dans le cadre d'un automate déterministe, un mot est accepté si le calcul, unique s'il existe, de l'automate dont il est l'étiquette est acceptant). Ce choix existentiel se produit au niveau d'un état de contrôle q et à la lecture d'une lettre a, pour lesquels $\delta(q,a)$ possède au moins deux éléments. Une variante possible de la condition d'acceptation (qui ne changerait rien au pouvoir de reconnaissance) serait de demander que tous les calculs d'étiquette u soient acceptants pour que u soit accepté. Le caractère existentiel est alors remplacé par un caractère universel. Dans le cas existentiel, il s'agit de deviner un calcul acceptant alors que dans le cas universel, il s'agit de tester tous les calculs.

Une solution combinant ces deux modèles est fournie par les automates alternants. Un automate alternant ne diffère d'un automate non déterministe que par sa fonction de transition, qui n'est plus à valeur dans les parties de Q, mais dans l'ensemble $\mathcal{B}^+(Q)$ des formules booléennes positives[1] sur Q. Par exemple $\delta(p,a) = q \vee (r \wedge s)$ signifie que dans un état p en lisant la lettre a, on va aller soit dans l'état q, soit poursuivre deux calculs en parallèle depuis les états r et s. Ainsi, un calcul ne sera plus une suite, linéaire, de transitions mais un arbre. Les automates non déterministes sont un cas particulier d'automates alternants, où δ ne prend pour valeur que des disjonctions d'états.

On dira qu'un sous-ensemble d'états $R \subseteq Q$ satisfait une formule booléenne Φ, ce que l'on notera $R \models \Phi$ si la valuation sur les états, où tous les éléments de R sont pris égaux à 1 et tous les éléments de $Q \setminus R$ sont pris égaux à 0, vérifie Φ.

On a la définition formelle suivante d'un *automate alternant*.

Définition 1.8 *Un* automate alternant *est un quintuplet $\mathcal{A} = \langle Q,A,I,F,\delta\rangle$, où Q est un ensemble fini d'états, A est un alphabet fini, $I \subseteq Q$ est un sous-ensemble d'états de Q appelé ensemble des états initiaux, $F \subseteq Q$ est un sous-ensemble de Q appelé ensemble des états finaux, et δ est une application de $Q \times A$ dans $\mathcal{B}^+(Q)$, appelée fonction de transition.*

1. Une formule booléenne positive est une formule dans laquelle la négation n'intervient pas, et où tout littéral est positif, c'est-à-dire égal à une variable (et non à sa négation). Par exemple la formule $p \vee (q \wedge r)$ est positive, ce qui n'est pas le cas de $p \vee \neg(q \wedge \overline{r})$.

Précisons comment est défini un arbre de calcul d'un automate alternant.

Définition 1.9 *Considérons un automate alternant $\mathcal{A} = \langle Q,A,I,F,\delta\rangle$. Considérons un mot $u = a_1a_2\cdots a_n$ sur l'alphabet A. Un arbre de calcul de $\mathcal{A}$ depuis un état p sur u est un arbre vérifiant les propriétés suivantes :*

- *la racine de l'arbre est étiquetée par p.*
- *si $n = 0$, la racine n'a pas de fils.*
- *si $n > 0$:*
  - *si $\delta(p,a_1) = ff$, la racine n'a qu'un fils dont l'étiquette est ff.*
  - *si $\delta(p,a_1) = tt$, la racine n'a qu'un fils dont l'étiquette est tt.*
  - *sinon, il existe $R = \{r_1,\ldots,r_{|R|}\} \subseteq Q$ tel que $R \models \delta(p,a_1)$, et la racine a $|R|$ fils, qui sont respectivement des arbres de calcul de $\mathcal{A}$ depuis $r_1,\ldots,r_{|R|}$ sur $a_2a_3\cdots a_n$.*

Enfin, un arbre de calcul *de $\mathcal{A}$ sur un mot u est un arbre de calcul de $\mathcal{A}$ depuis un état initial sur u.*

Précisons à présent comment on définit les mots acceptés par un automate alternant.

Définition 1.10 *Considérons un automate alternant $\mathcal{A} = \langle Q,A,I,F,\delta\rangle$ et un mot u sur l'alphabet A. Un arbre de calcul de $\mathcal{A}$ sur u est* acceptant *si toute feuille est étiquetée, soit par tt, soit par un état final. Un mot u est accepté par un automate alternant $\mathcal{A}$ s'il existe un arbre de calcul acceptant de $\mathcal{A}$ sur u. On note $L(\mathcal{A})$ l'ensemble des mots acceptés par $\mathcal{A}$.*

L'alternance ne rajoute pas de pouvoir aux automates finis.

Théorème 1.3 *Soit $\mathcal{A}$ un automate alternant et soit $L(\mathcal{A})$ le langage reconnu par $\mathcal{A}$. Il existe un automate non déterministe $\mathcal{A}'$ qui reconnaît le même langage, c'est-à-dire que l'on a $L(\mathcal{A}) = L(\mathcal{A}')$.*

Exemple 1.5 Considérons l'automate alternant $\mathcal{A} = \langle Q,A,I,F,\delta\rangle$ suivant :

- $Q = \{p,q,r\}$.
- $A = \{a,b\}$.
- $I = \{p\}$.
- $F = \{p\}$.
- δ est donnée par :
  - $\delta(p,a) = (r \wedge q) \vee p$.
  - $\delta(q,a) = p \vee r$.
  - $\delta(r,a) = r$.
  - $\delta(p,b) = r \vee q$.
  - $\delta(q,b) = r \vee p$.
  - $\delta(r,b) = p \wedge q$.

L'arbre donné dans la figure 1.4 est un arbre de calcul acceptant de $\mathcal{A}$ sur le mot $aaba$.

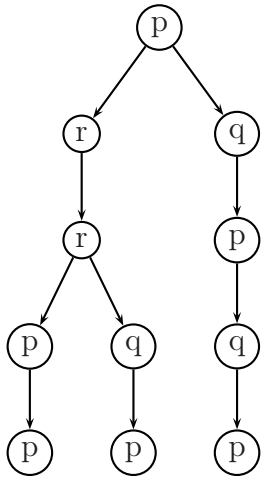


FIG. 1.4 – *Arbre de calcul acceptant de l'automate alternant de l'exemple 1.5*

Etant donné un automate alternant $\mathcal{A} = \langle Q,A,I,F,\delta\rangle$, on définit l'automate dual $\overline{\mathcal{A}} = \langle Q,A,I,Q \setminus F,\overline{\delta}\rangle$ de $\mathcal{A}$, où pour tout état q et toute lettre a, $\overline{\delta}(q,a)$ est la formule obtenue à partir de $\delta(q,a)$ en remplaçant tous les symboles $\vee$ par $\wedge$ et tous les symboles $\wedge$ par $\vee$, et en remplaçant tt par ff et ff par tt. On a alors facilement le résultat suivant.

Proposition 1.1 *Le langage reconnu par $\overline{\mathcal{A}}$ est le complémentaire du langage reconnu par $\mathcal{A}$.*

En particulier, la procédure de dualisation permet d'obtenir pour tout automate alternant un automate de même taille reconnaissant le langage complémentaire.

$\mathcal{P}$-automates

L'objet principal de cette thèse est de calculer des ensembles de positions gagnantes dans des jeux sur des graphes d'automates à pile. N'ayant pas encore défini ni les jeux ni les graphes d'automate à pile, disons que les noeuds d'un graphe d'automate à pile sont étiquetés par des couples formés d'un état de contrôle et d'un mot, c'est-à-dire qu'il s'agit d'éléments dans un ensemble de la forme $\mathcal{Q} \times \Gamma^*$, où $\mathcal{Q}$ est un ensemble fini d'états et Γ est un alphabet fini. L'ensemble des positions gagnantes pour un joueur est donc une partie de $\mathcal{Q} \times \Gamma^*$. Les automates, nous venons de le voir, permettent de représenter des langages rationnels et fournissent une procédure simple pour décider si un élément donné appartient au langage représenté par l'automate.

Une représentation naturelle d'une partie P de $\mathcal{Q} \times \Gamma^*$ est donnée par $(L_q)_{q\in\mathcal{Q}}$ où $L_q = \{u \in \Gamma^* \mid (q,u) \in P\}$. Dès lors, si L_q est rationnel pour tout $q \in \mathcal{Q}$, la partie P peut être représentée par un ensemble fini d'automates $(\mathcal{A}_q)_{q\in\mathcal{Q}}$. Dans ce cas, on dira que l'ensemble P est *régulier*.

Les $\mathcal{P}$-automates sont des objets naturels pour reconnaître de tels ensembles. Nous les définissons dans le cadre le plus général, à savoir celui de l'alternance.

Définition 1.11 *Un* $\mathcal{P}$-automate alternant *est un quintuplet $\mathcal{A} = \langle Q,\Gamma,I,F,\delta\rangle$, où*

Q est un ensemble fini d'états, A est un alphabet fini, $I \subseteq Q$ est un sous-ensemble d'états de Q appelé ensemble des états initiaux, $F \subseteq Q$ est un sous-ensemble de Q appelé ensemble des états finaux, et δ est une application de $Q \times \Gamma$ dans $\mathcal{B}^+(Q)$, appelée fonction de transition.

La différence vient du fait qu'un $\mathcal{P}$-automate accepte des couples de la forme (q,u) où $q \in I$ et $u \in \Gamma^*$. Pour cela, le calcul de l'automate alternant commence en i et ensuite se comporte normalement. Ainsi, la première composante du couple à reconnaître détermine l'état initial du calcul qui se déroule ensuite sur la seconde composante.

Définition 1.12 *Considérons un $\mathcal{P}$-automate alternant $\mathcal{A} = \langle Q,\Gamma,I,F,\delta\rangle$. Considérons un mot $u = a_1a_2\cdots a_n$ sur l'alphabet Γ et un état $q \in I$. Un arbre de calcul de $\mathcal{A}$ depuis un état p sur u est un arbre vérifiant les propriétés suivantes :*

- *la racine de l'arbre est étiquetée par p.*
- *si $n = 0$, la racine n'a pas de fils.*
- *si $n > 0$:*
  - *si $\delta(p,a_1) = ff$, la racine n'a qu'un fils dont l'étiquette est ff.*
  - *si $\delta(p,a_1) = tt$, la racine n'a qu'un fils dont l'étiquette est tt.*
  - *sinon, il existe $R = \{r_1,\ldots,r_{|R|}\} \subseteq Q$ tel que $R \models \delta(p,a_1)$, et la racine à $|R|$ fils, qui sont respectivement des arbres de calcul de $\mathcal{A}$ depuis $r_1,\ldots,r_{|R|}$ sur $a_2a_3\cdots a_n$.*

Enfin, un arbre de calcul *de $\mathcal{A}$ sur le couple (q,u) est un arbre de calcul de $\mathcal{A}$ depuis l'état initial q.*

Précisons enfin comment sont définies les configurations acceptées par un $\mathcal{P}$-automate alternant.

Définition 1.13 *Considérons un $\mathcal{P}$-automate alternant $\mathcal{A} = \langle Q,\Gamma,I,F,\delta\rangle$ et un couple $(q,u) \in I \times \Gamma^*$. Un arbre de calcul de $\mathcal{A}$ sur (q,u) est* acceptant *si toute feuille est étiquetée soit par tt soit par un état final. Un couple (q,u) est accepté par un automate alternant $\mathcal{A}$ s'il existe un arbre de calcul acceptant de $\mathcal{A}$ sur (q,u). On note $L(\mathcal{A})$ l'ensemble des couples acceptés par $\mathcal{A}$. Une partie P de $I \times A^*$ est reconnue par $\mathcal{A}$ si tout couple (q,u) de P est accepté par $\mathcal{A}$.*

Il est alors facile d'établir le résultat suivant.

Proposition 1.2 *Soit $\mathcal{Q}$ un ensemble fini et soit Γ un alphabet fini. L'ensemble des parties régulières de $\mathcal{Q} \times \Gamma^*$ est exactement l'ensemble des parties reconnues par des $\mathcal{P}$-automates alternants (ou non déterministes).*

1.2.2 Langages algébriques de mots finis

Considérons un alphabet fini $\mathcal{A}$ et le langage L défini par $L = \{u\widetilde{u} \mid u \in A^*\}$. Il n'est pas très difficile de voir qu'un tel langage ne peut être reconnu par un automate fini. Pour pouvoir reconnaître un tel langage, il faudrait se souvenir de toutes les lettres déjà lues, deviner quand on va commencer à lire le miroir et vérifier lettre

par lettre que celui-ci est correct. Dès lors, une mémoire non bornée (que n'offrent pas les automates finis) est nécessaire pour reconnaître L. Le modèle naturel est alors celui d'un automate fini auquel on adjoint une pile. On parle alors *d'automate à pile*. Les langages associés sont les *langages algébriques*.

Les langages algébriques ont été historiquement introduits à l'aide des grammaires algébriques. Nous n'en parlerons pas ici, puisque nous sommes essentiellement intéressés par les machines sous-jacentes, à savoir les automates à pile. Pour plus de détails sur les grammaires algébriques, on pourra consulter [9, 44].

Automate à pile

Un automate à pile n'est pas autre chose qu'un automate fini auquel on adjoint une pile qui restreint le champs d'action du contrôle fini. Plus formellement, on a la définition ci-dessous.

Définition 1.14 (Automate à pile) *Un automate à pile $\mathcal{A}$ est un septuplet $\langle Q,A,\Gamma,\bot,I,F,\Delta\rangle$, où Q est un ensemble fini d'états, A est un alphabet fini d'entrée, Γ est un alphabet fini de pile, $\bot$ est un symbole de Γ appelé symbole de fond de pile, $I \subseteq Q$ est un sous-ensemble de Q appelé ensemble des états initiaux, $F \subseteq Q$ est un sous-ensemble de Q appelé ensemble des états finaux, et Δ est une application de $Q \times \Gamma \times A$ dans $\mathcal{P}(\{skip(q),pop(q),push(q,\gamma) \mid q \in Q, \gamma \in \Gamma \setminus \{\bot\}\})$, appelée fonction de transition et telle que $\delta(p,\bot,a) \subseteq \{push(q,\gamma),skip(q) \mid q \in Q, \gamma \in \Gamma \setminus \{\bot\}\}$ pour tout $p \in Q$ et tout $a \in A$.*

Dans le cas des automates finis, un calcul se déroule de la façon suivante : on choisit un état initial, puis on lit la première lettre, et l'on change alors d'état en fonction de l'état précédent et de la lettre lue, et ainsi de suite jusqu'à avoir lu le mot en entier. Le calcul est alors acceptant si l'état courant est final. Dans le cas des automates à pile, le processus est similaire, excepté que l'on met (éventuellement) à jour la pile en empilant (règle de type *push*) ou en dépilant (règle de type *pop*) un symbole. De plus, le symbole en sommet de pile, la lettre lue, et l'état de contrôle, décident des actions possibles à effectuer tant au niveau du changement de l'état de contrôle que de l'actualisation de la pile. Ainsi, la notion importante pour définir un calcul n'est plus celle d'état de contrôle mais celle de *configuration*

Définition 1.15 *Soit $\mathcal{A}$ un automate à pile. Soit Q son ensemble d'états et soit Γ son alphabet de pile. L'ensemble des* configurations *de l'automate à pile est $Q \times (\Gamma \setminus \{\bot\})^*\bot$. Autrement dit, une configuration de $\mathcal{A}$ est un couple $(q,\sigma\bot)$ formé d'un état de contrôle q et d'un mot de pile $\sigma\bot$ contenant une unique occurrence du symbole de fond de pile. En terme de machine, une configuration $(q,\sigma\bot)$ représente un état de l'automate, où ce dernier est dans l'état q, et contient dans sa pile le mot $\sigma\bot$ si on lit cette dernière de haut en bas.*

Enfin, une configuration est initiale *si son état de contrôle est initial. De même, une configuration est* finale *si son état de contrôle est final.*

Définissons maintenant la notion de *calcul*

Définition 1.16 *Soit une configuration* $(q,\gamma\sigma)$ *de l'automate à pile, et soit* $a \in A$ *une lettre. A la lecture de* a, $\mathcal{A}$ *peut passer de la configuration* $(q,\gamma\sigma)$ *aux configurations suivantes :*

- $(q',\gamma\sigma)$ *pour tout* $skip(q') \in \Delta(q,\gamma,a)$.
- (q',σ) *pour tout* $pop(q') \in \Delta(q,\gamma,a)$.
- $(q',\gamma'\gamma\sigma)$ *pour tout* $push(q',\gamma') \in \Delta(q,\gamma,a)$.

On dit alors qu'il y a une transition d'étiquette a *de la première configuration* C *à la seconde* C', *ce que l'on note* $C \xrightarrow{a} C'$. *Un calcul* c *:* $C_0 \xrightarrow{a_1} C_1 \xrightarrow{a_2} C_2 \cdots C_{n-1} \xrightarrow{a_n} C_n$ *est une suite de transitions telles que l'origine de chacune coïncide avec l'extrémité de la précédente. Le mot* $a_1a_2\cdots a_n$ *est appelé étiquette du calcul* c. *Enfin, un calcul acceptant est un calcul dont l'origine est une configuration initiale et dont l'extrémité est une configuration finale.*

On peut alors définir le langage de mots finis associé à un automate à pile.

Définition 1.17 *Un mot fini* u *est* accepté *(ou reconnu) par un automate à pile* $\mathcal{A}$ *s'il est l'étiquette d'au moins un calcul acceptant de* $\mathcal{A}$. *On note* $L(\mathcal{A})$ *l'ensemble des mots acceptés par l'automate à pile* $\mathcal{A}$, *et* $L(\mathcal{A})$ *est appelé* langage reconnu par $\mathcal{A}$.

Un langage reconnu par un automate à pile est qualifié d'algébrique.

Sous-classes

On considère plusieurs sous-classes d'automates à pile et de leurs langages associées. La première est celle des automates à pile déterministes.

Définition 1.18 (Automate à pile déterministe temps-réel) *Un* automate à pile
$\mathcal{A} = \langle Q,A,\Gamma,\perp,I,F,\Delta\rangle$ *est* déterministe *si et seulement si* I *est un singleton et pour tout* $q \in Q$, $\gamma \in \Gamma$, $a \in A$, $\Delta(q,\gamma,a)$ *contient au plus un élément. Les langages reconnus par des automates à pile déterministes sont qualifiés* d'algébriques déterministes temps réel.

Remarque 1.2 Il existe une notion plus générale que les automates à pile déterministe temps-réel dans laquelle on autorise des ε-transitions. Dans ce modèle, l'automate peut faire des transitions sans lire de lettre en entrée. On parle alors d'automate à pile déterministe. Nous n'utiliserons pas ici ce modèle et nous nous autoriserons l'abus de langage consistant à ommetre la mention *temps réel* lorsque nous parlerons d'automates à pile déterministes ou de langages algébriques déterministes.

Si l'on reprend l'exemple précédent du langage algébrique $L = \{u\widetilde{u} \mid u \in A^*\}$, on voit facilement que L n'est pas algébrique déterministe puisqu'il faut intuitivement deviner pour un mot $w = u\widetilde{u}$ où se termine u et où commence $\widetilde{u}$. En revanche le langage $L' = \{u\#\widetilde{u} \mid u \in A^*\}$, où $\# \notin A$, est algébrique déterministe.

Une autre restriction concernant les automates à pile consiste à réduire l'alphabet de pile à deux lettres $\{a,\perp\}$. Dans ce cas, la pile agit comme un compteur sur lequel on peut effectuer le test à zéro.

Définition 1.19 (Automate à compteur) *Un automate à pile est un* automate à compteur *si et seulement si son alphabet de pile est réduit à deux lettres.*

Les langages L et L' précédents ne peuvent pas être reconnus par un automate à compteur. En revanche le langage L'' sur l'alphabet $A = \{a,b,c\}$ défini par $L'' = A^* \setminus \{a^n b^n c^n \mid n \geq 1\}$ est reconnu par un automate à compteur alors qu'il ne peut pas l'être par un automate à pile déterministe.

En résumé, on a le résultat ci-dessous.

Proposition 1.3 *La classe des langages reconnus par des automates à pile déterministe et la classe des langages reconnus par des automates à compteur sont incomparables.*

Langages Vpl

On décrit enfin une notion récemment introduite par R. Alur et P. Madhusudan dans [4], celle de *langage* Vpl (pour Visibly Pushdown Language). Les langages Vpl forment une sous-classe stricte des langages reconnus par des automates à pile déterministes.

De tels langages sont définis sur des alphabets *d'actions*.

Définition 1.20 *Un* alphabet d'actions *est un triplet* $\widetilde{A} = \langle A_c,A_r,A_{int}\rangle$ *formé de trois alphabets disjoints, où* A_c *est un alphabet fini* d'appels*,* A_r *est un alphabet fini* de retours*, et* A_{int} *est un alphabet fini* d'actions internes*. Pour un tel alphabet* $\widetilde{A}$*, on note* $A = A_c \cup A_r \cup A_{int}$*.*

On définit alors la notion *d'automate à pile avec visibilité*, ou Vpa

Définition 1.21 *Un* automate à pile avec visibilité*, ou* Vpa*, sur un alphabet d'actions* $\langle A_c,A_r,A_{int}\rangle$*, est un automate à pile* $\langle Q,A,\Gamma,\perp,I,F,\Delta\rangle$ *tel que :*

- *pour tout appel* $a \in A_c$*, pour tout état* $q \in Q$ *et tout sommet de pile* $\gamma \in \Gamma$*,* $\Delta(q,\gamma,a) \subseteq \{push(q',\gamma') \mid q' \in Q \text{ et } \gamma' \in \Gamma\}$*. De plus,* $\Delta(q,\gamma,a)$ *ne dépend pas de* γ*, c'est-à-dire que* $\Delta(q,\gamma,a) = \Delta(q,\gamma',a)$ *pour tout* $q \in Q$*,* $\gamma,\gamma' \in \Gamma$ *et* $a \in A_c$*.*
- *pour tout retour* $a \in A_r$*, pour tout état* $q \in Q$ *et tout sommet de pile* $\gamma \in \Gamma \setminus \{\perp\}$*,* $\Delta(q,\gamma,a) \subseteq \{pop(q') \mid q' \in Q\}$*. Pour tout retour* $a \in A_r$*, pour tout état* $q \in Q$*,* $\Delta(q,\perp,a) \subseteq \{skip(q') \mid q' \in Q\}$*.*
- *pour tout action interne* $a \in A_{int}$*, pour tout état* $q \in Q$ *et tout sommet de pile* $\gamma \in \Gamma$*,* $\Delta(q,\gamma,a) \subseteq \{skip(q') \mid q' \in Q\}$*. De plus,* $\Delta(q,\gamma,a)$ *ne dépend pas de* γ*, c'est-à-dire que* $\Delta(q,\gamma,a) = \Delta(q,\gamma',a)$ *pour tout* $q \in Q$*,* $\gamma,\gamma' \in \Gamma$ *et* $a \in A_{int}$*.*

Notation 1.3 *Par définition, lorsque a est un appel ou une action interne,* $\Delta(q,\gamma,a)$ *ne dépend pas de* γ*. Ainsi, lorsque* γ *n'a pas de signification particulière, on notera* $\Delta(q,\_,a)$ *à la place de* $\Delta(q,\gamma,a)$*.*

Ainsi, un VPA empile lorsqu'il lit un appel, dépile lorsqu'il lit un retour, et ne modifie pas sa pile lorsqu'il lit une action interne. De plus le sommet de pile n'intervient que dans le choix d'une règle de dépilement.

Un langage reconnu par un VPA est qualifié de *langage* VPL. La classe des langages VPL, outre son utilité pour la vérification, possèdent de nombreuses propriétés qui sont présentées dans [4]. Signalons l'une d'entre elles.

Proposition 1.4 *[4] Pour tout* VPA *non déterministe, il existe un* VPA *déterministe qui reconnaît le même langage.*

1.2.3 Langages rationnels de mots infinis

A la manière de ce que l'on a fait pour les langages de mots finis, on définit les langages rationnels de mots infinis comme le plus petit ensemble contenant des langages de base et fermé pour un certains nombres d'opérations. Pour cela, on commence comme pour les langages de mots finis, sauf qu'on va en plus demander la stabilité par l'opération $L \mapsto L^\omega$. Dès lors, la classe obtenue contient des langages de mots finis et infinis. Les langages rationnels de mots infinis sont alors définis comme les langages de mots infinis appartenant à la classe précédente.

Définition 1.22 *Etant donné un alphabet A, la classe des* langages ∞-rationnels *sur A est la plus petit classe de langages $Rat\ A^\infty$ satisfaisant les propriétés suivantes :*

- $\emptyset \in Rat\ A^\infty$.
- $\{a\} \in Rat\ A^\infty$ *pour toute lettre* $a \in A$.
- $Rat\ A^\infty$ *est fermé pour l'union.*
- *pour tout langage de mots finis L et pour tout langage de mots finis ou infinis L', si L,$L' \in Rat\ A^\infty$ alors $LL' \in Rat\ A^\infty$.*
- *pour tout langage de mots finis L, si $L \in Rat\ A^\infty$ alors $L^* \in Rat\ A^\infty$ et $L^\omega \in Rat\ A^\infty$.*

Enfin, la classe des langages ω-rationnels *sur A est l'ensembles des langages de mots infinis appartenant à $Rat\ A^\infty$, c'est-à-dire que $Rat\ A^\omega = Rat\ A^\infty \cap \mathcal{P}(A^\omega)$.*

Remarque 1.3 On a également que $Rat\ A^*$ est l'ensemble des langages de mots finis appartenant à $Rat\ A^\infty$, c'est-à-dire que $Rat\ A^* = Rat\ A^\infty \cap \mathcal{P}(A^*)$.

Comme dans le cas des langages rationnels, on utilise les *expressions rationnelles* pour représenter les langages ∞-rationnels (et ω-rationnels).

Définition 1.23 *Les expressions ∞-rationnelles et leur langage ∞-rationnel associé, sont définis récursivement :*

- $\emptyset$ *est une expression ∞-rationnelle et est associée au langage ∞-rationnel $\emptyset$.*
- a *est une expression ∞-rationnelle et est associée au langage ∞-rationnel $\{a\}$ pour toute lettre $a \in A$.*
- ε *est une expression ∞-rationnelle et est associée au langage ∞-rationnel $\{\varepsilon\}$.*
- *si E_1 et E_2 sont des expressions ∞-rationnelles respectivement associées aux langages ∞-rationnels L_1 et L_2, $(E_1 \cup E_2)$ est une expression ∞-rationnelles associée au langage ∞-rationnel $L_1 \cup L_2$.*

– *si E_1 et E_2 sont des expressions ∞-rationnelles respectivement associées aux langages ∞-rationnels L_1 et L_2 et si de plus E_1 ne contient pas de $^\omega$ (c'est-à-dire que L_1 est un langage de mots finis), $(E_1 \cdot E_2)$, E_1^* et E_1^ω sont des expressions ∞-rationnelles respectivement associées aux langages ∞-rationnels $L_1 \cdot L_2$, L_1^* et L_1^ω.*

Lorsqu'il n'y a pas d'ambiguïté, on supprime les parenthèses inutiles.

Dans la suite, on se permettra l'abus de langage consistant à confondre une expression ∞-rationnelle avec le langage qu'elle représente. Enfin, les expressions ∞-rationnelles généralisant les expressions rationnelles, on les confondra avec ces dernières.

Automates de Büchi et langages reconnaissables

Comme dans le cadre des mots finis, on propose un modèle de machines reconnaissant exactement les langages rationnels : les *automates de Büchi*. Les automates de Büchi fonctionnent comme les automates pour les mots finis, sauf en ce qui concerne la procédure d'acceptation. En effet, l'acceptation par état final n'est plus très pertinente dans le cas des mots infinis puisque la lecture d'un tel mot ne se termine jamais. Cependant, le nombre d'états étant fini, lors de la lecture d'un mot infini, certains états vont être infiniment souvent visités. Dès lors, le critère adopté pour décider si un calcul est acceptant ou non n'est plus le dernier état atteint, mais l'ensemble des états infiniment souvent visités au cours de la lecture du mot infini par l'automate.

Plus formellement, on a les définitions suivantes.

Définition 1.24 (Automate de Büchi) *Un* automate de Büchi $\mathcal{A}$ *est un quintuplet $\langle Q,A,I,F,\delta\rangle$, où Q est un ensemble fini d'états, A est un alphabet fini, $I \subseteq Q$ est un sous-ensemble de Q appelé ensemble des états initiaux, $F \subseteq Q$ est un sous-ensemble de Q appelé ensemble des états finaux, et δ est une application de $Q \times A$ dans $\mathcal{P}(Q)$, appelée fonction de transition.*

Définition 1.25 *Etant donné un automate de Büchi $\mathcal{A} = \langle Q,A,I,F,\delta\rangle$, on dit qu'il y a une transition d'un état p à un état q d'étiquette a, ce que l'on note $p \xrightarrow{a} q$ si $q \in \delta(p,a)$. Un* calcul *$c : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_n} q_3 \cdots$ est une suite de transitions telles que l'origine de chacune coïncide avec l'extrémité de la précédente. Le mot $a_1a_2a_3\cdots$ est appelée* l'étiquette *du calcul c. L'état q_0 est appelé* origine *de c. Lorsque le calcul c est infini, on note $Inf(c)$ l'ensemble des états qui apparaissent infiniment souvent dans c, c'est-à-dire $Inf(c) = \{q \in Q \mid \exists^\infty i \text{ t.q. } q_i = q\}$. Enfin, un calcul acceptant est un calcul infini c dont l'origine est un état initial et tel que $Inf(c) \cap F \neq \emptyset$.*

Définition 1.26 *Un mot infini u est* accepté *(ou reconnu) par un automate de Büchi $\mathcal{A}$ s'il est l'étiquette d'au moins un calcul acceptant de $\mathcal{A}$. On note $L(\mathcal{A})$ l'ensemble des mots acceptés par l'automate $\mathcal{A}$, et $L(\mathcal{A})$ est appelé* langage reconnu par $\mathcal{A}$.

La représentation graphique d'un automate de Büchi est la même que celle d'un automate pour les mots finis, à la seule différence que les états finaux ne possèdent plus de flèche sortante mais sont doublement cerclés.

Exemple 1.6 Soit $\mathcal{A} = \langle Q,A,I,F,\delta \rangle$ l'automate défini par :
- $Q = \{p,q,r\}$.
- $A = \{a,b\}$.
- $I = \{p\}$.
- $F = \{q\}$.
- $\delta(p,a) = \{p\}$, $\delta(p,b) = \{p,q\}$, $\delta(q,a) = \{r\}$, $\delta(q,b) = \{q\}$, $\delta(r,a) = \{r\}$ et $\delta(r,b) = \{r\}$.

La représentation graphique de cet automate est donnée par la figure 1.5. Le langage reconnu par $\mathcal{A}$ est l'ensemble des mots ne contenant qu'un nombre fini de a, c'est-à-dire que $L(\mathcal{A}) = \{u \in \{a,b\}^\omega \mid |u|_a < \infty\}$.

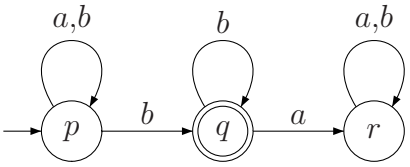


FIG. 1.5 – *Représentation graphique d'un automate de Büchi*

Les automates de Büchi fournissent donc un autre moyen de représenter des langages de mots infinis. Un langage reconnu par un automate fini est qualifié de *reconnaissable*

Définition 1.27 *Un langage de mots infinis est* ω-reconnaissable *s'il existe un automate de Büchi qui le reconnaît. Etant donné un alphabet A, on note Rec A^ω l'ensemble des langages ω-reconnaissables sur l'alphabet A.*

Quel est le lien entre les langages ω-rationnels de mots infinis et les langages ω-reconnaissables de mots infinis sur un alphabet fini A? La réponse nous est donnée par le théorème de Kleene Büchi

Théorème 1.4 (Kleene Büchi) *Pour tout alphabet fini A, les langages ω-rationnels de A et les langages ω-reconnaissables de A coïncident :*

$$Rat\ A^\infty = Rec\ A^\infty$$

Automates de Muller et automates de parité

Dans le cas des mots finis, nous avions définis les automates déterministes, et noté que ceux-ci reconnaissent les mêmes langages que leurs homologues non déterministes. De la même façon que pour les mots finis, on peut définir les automates de

Büchi déterministes. Cependant, ces derniers ne reconnaissant pas tous les langages ω-réguliers.

Proposition 1.5 *Le langage ω-régulier $\{u \in \{a,b\}^\omega \mid |u|_a < \infty\}$ n'est pas reconnaissable par un automate déterministe de Büchi.*

L'intérêt d'un modèle de calcul déterministe par rapport à un modèle non déterministe apparaîtra naturellement au cours de cette thèse, par exemple lorsque nous ferons des *produits* entre des graphes et des automates. La faiblesse des automates déterministes de Büchi vient certes de leur caractère prévisible, mais aussi de leur condition d'acceptation qui a un champ d'action assez limité, puisqu'elle ne permet de distinguer que deux types d'états, les bons et les mauvais. On pourrait en effet autoriser des mauvais comportements du moment que d'autres se produisent, c'est-à-dire hiérarchiser les événements et demander que le plus petit état infiniment souvent répété soit dans un certain ensemble. La condition de *parité* permet cela : on attribue à chaque état un entier, que l'on appelle couleur, et parmi les états apparaissant infiniment souvent lors d'un calcul, on regarde celui qui a la plus petite couleur. Si cette couleur est paire, le calcul est acceptant, sinon il ne l'est pas. Plus formellement on a la définition suivante.

Définition 1.28 (Automate de parité) *Un* automate de parité $\mathcal{A}$ *est un quintuplet $\langle Q,A,I,\delta,\rho\rangle$, où Q est un ensemble fini d'états, A est un alphabet fini, $I \subseteq Q$ est un sous-ensemble de Q appelé ensemble des états initiaux, δ est une application de $Q \times A$ dans $\mathcal{P}(Q)$, appelée fonction de transition, et ρ est une application de* coloriage *de Q dans un ensemble fini $C \subset \mathbb{N}$ de* couleurs.

Définition 1.29 *Etant donné un automate de parité $\mathcal{A} = \langle Q,A,I,\delta,\rho\rangle$, on dit qu'il y a une transition d'un état p à un état q d'étiquette a, ce que l'on note $p \xrightarrow{a} q$ si $q \in \delta(p,a)$. Un* calcul $c : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_n} q_3 \cdots$ *est une suite de transitions telles que l'origine de chacune coïncide avec l'extrémité de la précédente. Le mot $a_1a_2a_3\cdots$ est appelée* l'étiquette *du calcul c. L'état q_0 est appelé* origine *de c. A un calcul infini $c = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_n} q_3 \cdots$, on associe un mot $\rho(c)$ sur l'alphabet C, en posant $\rho(c) = c_1c_2\cdots$ avec $c_i = \rho(q_i)$. On note $Inf(\rho(c))$ l'ensemble des couleurs qui apparaissent infiniment souvent dans $\rho(c)$, c'est-à-dire $Inf(\rho(c)) = \{c \in C \mid \exists^\infty i \text{ t.q. } \rho(q_i) = c\}$. Enfin, un calcul acceptant est un calcul c dont l'origine est un état initial et tel que* $\min(Inf(c))$ *est pair.*

Un mot infini u est accepté *(ou reconnu) par un automate de parité $\mathcal{A}$ s'il est l'étiquette d'au moins un calcul acceptant de $\mathcal{A}$. On note $L(\mathcal{A})$ l'ensemble des mots acceptés par l'automate $\mathcal{A}$, et $L(\mathcal{A})$ est appelé* langage reconnu par $\mathcal{A}$.

La représentation graphique d'un automate de parité est la même que celle d'un automate de Büchi, à ceci près qu'il n'y a pas d'états finaux donc pas d'états doublement cerclé, et que l'on écrit sa couleur à coté de l'étiquette de l'état.

Exemple 1.7 Considérons à nouveau le langage $L = \{u \in \{a,b\}^\omega \mid |u|_a < \infty\}$. Le langage L est reconnu par l'automate de parité $\mathcal{A} = \langle Q,A,I,\delta,C,\rho\rangle$ représenté par la figure 1.6 et défini comme suit :

– $Q = \{p,q\}$.

- $A = \{a,b\}$.
- $I = \{p\}$.
- $\delta(p,a) = \delta(q,a) = a$ et $\delta(p,b) = \delta(q,b) = b$.
- $\rho(p) = 1$ et $\rho(q) = 2$.

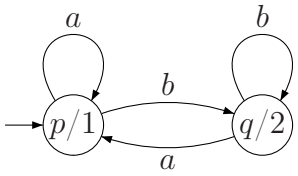


FIG. 1.6 – *Représentation graphique d'un automate de parité*

Remarque 1.4 Les automates de Büchi sont un cas particulier d'automate de parité à deux couleurs, 0 et 1. En effet, étant donné un automate de Büchi $\langle Q,A,I,F,\delta\rangle$, il est facile de voir que l'automate de parité $\langle Q,A,I,\rho,\delta\rangle$ où $\rho(q) = 0$ si $q \in F$ et $\rho(q) = 1$ sinon, reconnaît le même langage.

L'automate de parité donné dans l'exemple précédent utilise deux couleurs, 1 et 2. Il est qualifié d'automate de co-Büchi. Un calcul est acceptant s'il ne visite qu'un nombre fini d'états colorés par 1. La condition de co-Büchi peut être décrite en terme d'états interdits (qui sont ceux colorés par 1) : un calcul est acceptant si et seulement s'il ne visite que finiment souvent des états interdits.

Une autre condition d'acceptation, et donc une autre famille d'automates, est due à Muller.

Définition 1.30 (Automate de Muller) *Un* automate de Muller $\mathcal{A}$ *est un quintuplet* $\langle Q,A,I,\mathcal{F},\delta\rangle$*, où* Q *est un ensemble fini d'états,* A *est un alphabet fini,* $I \subseteq Q$ *est un sous-ensemble de* Q *appelé ensemble des états initiaux,* $\mathcal{F} \subseteq \mathcal{P}(Q)$ *est une collection de sous-ensembles de* Q *appelés ensembles terminaux, et* δ *est une application de* $Q \times A$ *dans* $\mathcal{P}(Q)$*, appelée fonction de transition.*

Définition 1.31 *Etant donné un automate de Muller* $\mathcal{A} = \langle Q,A,I,\mathcal{F},\delta\rangle$*, on dit qu'il y a une transition d'un état* p *à un état* q *d'étiquette* a*, ce que l'on note* $p \xrightarrow{a} q$ *si* $q \in \delta(p,a)$*. Un* calcul $c : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_n} q_3 \cdots$ *est une suite de transitions telles que l'origine de chacune coïncide avec l'extrémité de la précédente. Le mot* $a_1a_2a_3\cdots$ *est appelée* l'étiquette *du calcul* c*. L'état* q_0 *est appelé* origine *de* c*. On note* $Inf(c)$ *l'ensemble des états qui apparaissent infiniment souvent dans* c*, c'est-à-dire* $Inf(c) = \{q \mid \exists^\infty i \text{ t.q. } q_i = q\}$*. Enfin, un calcul acceptant est un calcul* c *dont l'origine est un état initial et tel que* $Inf(c) \in \mathcal{F}$*.*

Un mot infini u *est* accepté *(ou reconnu) par un automate de Muller* $\mathcal{A}$ *s'il est l'étiquette d'au moins un calcul acceptant de* $\mathcal{A}$*. On note* $L(\mathcal{A})$ *l'ensemble des mots acceptés par l'automate* $\mathcal{A}$*, et* $L(\mathcal{A})$ *est appelé* langage reconnu par $\mathcal{A}$*.*

La représentation graphique d'un automate de Muller est la même que celle d'un automate de Büchi, à ceci près qu'il n'y a pas d'états doublement cerclés.

Exemple 1.8 Considérons à nouveau le langage $L = \{u \in \{a,b\}^\omega \mid |u|_a < \infty\}$. Le langage L est reconnu par l'automate de Muller $\mathcal{A} = \langle Q,A,I,\mathcal{F},\delta\rangle$, représenté par la figure 1.7, et défini comme suit :

- $Q = \{p,q\}$.
- $A = \{a,b\}$.
- $I = \{p\}$.
- $\delta(p,a) = \delta(q,a) = a$ et $\delta(p,b) = \delta(q,b) = b$.
- $\mathcal{F} = \{\{q\}\}$.

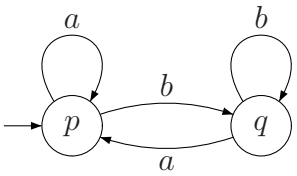


FIG. 1.7 – *Représentation graphique d'un automate de Muller*

Concernant le pouvoir de reconnaissance des automates de parité et de Muller, on a le résultat suivant [70]

Proposition 1.6 *Soit L un langage de mots infinis. Les propriétés suivantes sont équivalentes :*

- *il existe un automate de Büchi reconnaissant L;*
- *il existe un automate déterministe de parité reconnaissant L;*
- *il existe un automate déterministe de Muller reconnaissant L.*

Automates alternants sur les mots infinis

Signalons enfin que l'on peut également considérer des automates alternants pour reconnaître des mots infinis. Ces derniers sont définis comme précédemment, sauf que maintenant un arbre de calcul peut avoir des branches infinies. La condition d'acceptation doit alors être vérifiée pour toutes les branches infinies. Pour ce qui est des branches finies, on demande qu'elles se terminent par $\#$. Ainsi, pour une condition de Büchi, on demandera que toute branche infinie contiennent une infinité d'états finaux. Pour une condition de parité, on demandera que dans toute branche infinie, la plus petite couleur apparaissant infiniment souvent soit paire.

Il est à noter que l'on n'accroît pas le pouvoir de reconnaissance en ajoutant de l'alternance.

Proposition 1.7 *Soit L un langage de mots infinis. Les propriétés suivantes sont équivalentes :*

- *il existe un automate alternant de Büchi reconnaissant L;*

– *il existe un automate alternant de parité reconnaissant L;*
– *il existe un automate alternant de Muller reconnaissant L;*
– *il existe un automate de Büchi reconnaissant L.*

1.2.4 Langages algébriques de mots infinis

On propose maintenant un modèle d'automates à pile reconnaissant des langages de mots infinis. L'idée est la même que pour les automates sans pile : on considère l'ensemble des états apparaissant infiniment souvent au cours d'un calcul pour déterminer si ce dernier est acceptant. Un langage de mots infinis accepté par un automate à pile sera qualifié de ω-algébrique. Signalons que comme dans le cadre des mots finis, il existe une approche fondée sur les grammaires, à savoir les grammaires hors-contexte. Les automates à pile sur les mots infinis ont été introduits et caractérisés dans [26, 53]. Pour l'approche par des grammaires hors-contexte, on consultera [66, 67, 10].

Automates à pile de Büchi

Définition 1.32 (Automate à pile de Büchi) *Un automate à pile de Büchi $\mathcal{A}$ est un septuplet $\langle Q,A,\Gamma,\bot,I,F,\delta\rangle$, où Q est un ensemble fini d'états, A est un alphabet fini d'entrée, Γ est un alphabet fini de pile, $\bot$ est un symbole de Γ appelé symbole de fond de pile, $I \subseteq Q$ est un sous-ensemble de Q appelé ensemble des états initiaux, $F \subseteq Q$ est un sous-ensemble de Q appelé ensemble des états finaux, et δ est une application de $Q \times \Gamma \times A$ dans $\mathcal{P}(\{skip(q),pop(q),push(q,\gamma) \mid q \in Q, \gamma \in \Gamma \setminus \{\bot\}\})$, appelée fonction de transition et telle que $\delta(p,\bot,a) \subseteq \{push(q,\gamma),skip(q) \mid q \in Q, \gamma \in \Gamma \setminus \{\bot\}\}$ pour tout $p \in Q$ et tout $a \in A$.*

La notion de configuration se définit comme pour les automate à pile sur les mots finis. On donne la notion de calcul.

Définition 1.33 *Soit une configuration $(q,\gamma\sigma)$ de l'automate à pile et soit $a \in A$ une lettre. A la lecture de a, $\mathcal{A}$ peut passer de la configuration $(q,\gamma\sigma)$ aux configurations suivantes :*

– *$(q',\gamma\sigma)$ pour tout $skip(q') \in \delta(q,\gamma,a)$.*
– *(q',σ) pour tout $pop(q') \in \delta(q,\gamma,a)$.*
– *$(q',\gamma'\gamma\sigma)$ pour tout $push(q',\gamma') \in \delta(q,\gamma,a)$.*

On dit alors qu'il y a une transition d'étiquette a de la première configuration C à la seconde C', ce que l'on note $C \xrightarrow{a} C'$. Un calcul $c : C_0 \xrightarrow{a_1} C_1 \xrightarrow{a_2} C_2 \cdots$ est une suite de transitions telle que l'origine de chacune coïncide avec l'extrémité de la précédente. Le mot $a_1a_2\cdots$ est appelé l'étiquette du calcul c. Enfin, un calcul acceptant est un calcul infini dont l'origine est une configuration initiale et qui contient une infinité de configurations finales.

On peut alors définir le langage de mots infinis associé à un automate à pile de Büchi.

Définition 1.34 *Un mot infini u est* accepté *(ou reconnu) par un automate à pile de Büchi $\mathcal{A}$ s'il est l'étiquette d'au moins un calcul acceptant de $\mathcal{A}$. On note $L(\mathcal{A})$ l'ensemble des mots acceptés par l'automate à pile de Büchi $\mathcal{A}$, et $L(\mathcal{A})$ est appelé* langage reconnu par $\mathcal{A}$.

Un langage reconnu par un automate à pile de Büchi est qualifié de ω-algébrique.

Automates à pile de parité et de Muller

On peut également définir les automates à pile muni d'une condition de parité ou d'une condition de Muller. Dans le cas des conditions de parité, on se donne une application associant à chaque état une couleur. La couleur d'une configuration est alors la couleur de son état. Enfin, un calcul est acceptant si et seulement si la plus petite couleur apparaissant infiniment souvent est paire.

Pour ce qui est des conditions de Muller, on se donne une collection $\mathcal{F}$ de sous-ensembles d'états de contrôle. Etant donné un calcul, on regarde le sous-ensemble d'états apparaissant infiniment souvent dans la suite des configurations. Si ce dernier est dans $\mathcal{F}$ le calcul est acceptant, sinon il est rejetant.

Enfin, un langage accepté par un automate à pile déterministe muni d'une condition de parité ou de Muller sera qualifié de langage *ω-algébrique déterministe*. Les langages ω-algébriques déterministes forment une sous-classe stricte des langage ω-algébriques.

Langages ω-VPL

Comme dans le paragraphe 1.2.2, on considère des alphabets d'actions et l'on définit les VPA munis de condition de Büchi et de Muller de la même façon que l'on définit les automates à pile de Büchi et de Muller.

Concernant le pouvoir de reconnaissance, R. Alur et P. Madhusudan ont donné le résultat suivant.

Théorème 1.5 *[4] Pour tout* VPA *non déterministe de Muller, il existe un* VPA *non déterministe de Büchi qui reconnaît le même langage.*

Il existe un langage reconnu par un VPA *non déterministe de Büchi qui n'est reconnu par aucun* VPA *déterministe de Muller.*

On définira donc les langages ω-VPL comme ceux acceptés par des VPA non déterministes de Büchi. Un langage accepté par un VPA déterministe de Büchi sera quant à lui qualifié de langage ω-VPL *déterministe*

1.3 Quelques familles de graphes infinis finiment représentables

L'objet de cette thèse est l'étude de jeux sur des graphes infinis. L'approche étant résolument algorithmique, elle suppose une description finie des entrées, et en partie des graphes considérés. Dès lors, les graphes infinis considérés seront finiment représentables.

Pour cela, nous allons considérer les graphes associés à des automates à pile ainsi qu'à leurs variantes. La construction est simple : étant donné un automate à pile, les sommets du graphe associé sont les configurations de l'automate à pile, alors que les arcs du graphe correspondent à l'existence d'une transition entre deux configurations dans l'automate à pile.

1.3.1 Graphe associé à un processus à pile

Dans ce cadre, on considère une version édulcorée des automates à pile dans laquelle on supprime les concepts d'états initiaux et finaux. On parle alors de *processus à pile*

Définition 1.35 *Un processus à pile $\mathcal{P}$ est un quintuplet $\langle Q,A,\Gamma,\bot,\Delta\rangle$ où Q est un ensemble fini d'états, A est un alphabet fini d'entrée, Γ est un alphabet fini de pile, $\bot$ est un symbole de Γ appelé symbole de fond de pile et Δ est une application de $Q\times\Gamma\times A$ dans $\mathcal{P}(\{skip(q), pop(q), push(q,\gamma) \mid q \in Q, \gamma \in \Gamma\setminus\{\bot\}\})$ appelée fonction de transition et telle que $\delta(p,\bot,a) \subseteq \{push(q,\gamma), skip(q) \mid q \in Q, \gamma \in \Gamma \setminus \{\bot\}\}$, pour tout état $p \in Q$ et toute lettre $a \in A$.*

On appelle $Q \times (\Gamma \setminus \{\bot\})^*\bot$ l'ensemble des configurations *de $\mathcal{P}$. Enfin, comme dans la définition 1.16, on note par $C \xrightarrow{a} C'$ le fait qu'il y ait une transition d'étiquette a entre deux configurations C et C'.*

Etant donné un processus à pile, on lui associe un graphe infini de la façon suivante.

Définition 1.36 *Soit un processus à pile $\mathcal{P} = \langle Q,A,\Gamma,\bot,\Delta\rangle$, et soit V l'ensemble de ses configurations. Soit $\rightarrow$ la relation de transition introduite dans la définition 1.35. On considère la relation $E \subseteq V \times A \times V$ donnée par $E = \{(v,a,v') \mid v \xrightarrow{a} v'\}$. Le graphe A-étiqueté $G = (V,E)$ est le* graphe associé au processus à pile $\mathcal{P}$.

Si l'on souhaite ignorer l'étiquetage sur les arcs, il suffit de partir d'un processus à pile dont l'alphabet d'entrée est réduit à une lettre, qui du coup étiquette tous les arcs du graphe, et qui peut alors être ignorée. Dans ce cadre, on omettra l'alphabet d'entrée dans la définition du processus à pile sous-jacent. Lorsque le contexte est clair, nous ne préciserons pas systématiquement si l'on considère des processus à pile avec ou sans étiquetage.

1.3.2 Sous-classes

On considérera aussi par la suite trois cas particuliers : les *graphes de processus à compteur*, les *graphes de* Vpp, et les *graphes de* Bpa. Le premier correspond simplement aux graphes engendrés par des *processus à compteur*, c'est-à-dire des processus à pile dont l'alphabet de pile ne contient que deux symboles.

Le deuxième correspond quant à lui aux graphes engendrés par des Vpp, c'est-à-dire des processus à pile obtenus à partir de Vpa. En d'autres termes, un graphe de Vpp est un graphe de processus à pile dans lequel les arcs correspondant à une règle d'empilement sont étiquetés par des appels, les arcs correspondant à une règle

de dépilement sont étiquetés par des retours, et les arcs correspondant à une règle ne modifiant pas la pile sont étiquetés par des actions internes.

Pour les graphe de BPA, on commence par donner une variante des processus à pile sans entrée, les *Basic Process Algebra* (BPA) [7]:

Définition 1.37 (BPA) *Un processus à pile sans entrée est un* BPA *si et seulement s'il ne possède qu'un seul état. Ainsi, un* BPA *est-il un triplet* $\mathcal{B} = \langle \Gamma, \bot, \Delta \rangle$*, où* Γ *est un alphabet de pile de fond de pile* $\bot$*, et* Δ *est une application de* Γ *dans* $\mathcal{P}(\{skip, pop, push(\gamma) \mid \gamma \in \Gamma\})$*, appelée* fonction de transition *et telle que* $pop \notin \Delta(\bot)$.

Enfin, un *graphe de* BPA est un graphe engendré par un BPA.

Des exemples de graphes de processus à pile et de graphes de BPA seront donnés dans les chapitres suivants.

1.4 Machine de Turing, calculabilité et complexité

On rappelle maintenant quelques notions de calculabilité et de complexité. Il serait impossible d'être exhaustif sur ce sujet, et l'on se contente donc de donner les concepts utiles pour la suite. Pour plus de détails sur ces notions, on consultera par exemple [69, 82, 44].

On commence par rappeler la notion de machine de Turing. Informellement, il s'agit d'une machine possédant un contrôle fini et travaillant sur un ruban qui est infini à droite. Sur ce ruban est écrit un mot fini suivi d'une infinité de symboles blancs. La machine possède une tête de lecture qui pointe au départ sur la case la plus à gauche. Un calcul commence dans un état de contrôle particulier, l'état initial. Selon l'état de contrôle et la lettre sous la tête de lecture, la machine récrit le symbole courant, change d'état de contrôle, et peut éventuellement se déplacer d'une case vers la gauche ou vers la droite. Enfin, elle peut s'arrêter, signifiant ainsi que le calcul est terminé, et éventuellement que le mot est accepté ou rejeté. Plus formellement, on a les définitions suivantes (calquées sur [69]).

Définition 1.38 *Une* machine de Turing déterministe *est un quadruplet* $M = (Q, A, q_{in}, \delta)$*, où* Q *est un ensemble fini d'états,* $q_{in} \in Q$ *est l'état initial, et* A *est un alphabet fini, disjoint de* Q*. L'alphabet* A *contient toujours deux symboles particuliers,* $\sqcup$ *et* $\rhd$*: le* blanc *et le* marqueur de début*. Enfin, la* fonction de transition δ *est une application de* $Q \times A$ *dans* $(Q \cup \{yes, no, h\}) \times A \times \{\leftarrow, \rightarrow, -\}$*, où l'on suppose que l'état acceptant yes, l'état rejetant no, et l'état d'arrêt* h *ne sont pas dans* Q*, et que les symboles* $\leftarrow, \rightarrow, -$ *ne sont pas dans* $Q \cup A$.

La fonction de transition δ donne, pour tout état de contrôle $q \in Q$ et tout symbole courant $a \in A$, un triplet $\delta(q,a) = (q', a', D)$, où q' est le nouvel état de contrôle, a' le symbole qui doit être écrit à la place de a, et $D \in \{\leftarrow, \rightarrow, -\}$ la direction dans laquelle doit aller la tête de lecture. Si $D = \leftarrow$, la tête se déplace vers la gauche, si $D = \rightarrow$, la tête se déplace vers la droite, et si $D = -$ elle ne bouge pas. On demande de plus, pour tout état q, que $\delta(q, \rhd)$ soit toujours de la forme

$(q', \triangleright , \rightarrow)$: le marqueur de début n'est jamais effacé, et l'on ne va jamais à sa gauche.

Au départ, l'état de contrôle est q_{in}, et le ruban contient $\triangleright$ suivi d'un mot fini $x \in (A \setminus \{\sqcup\})^*$. On dit que x est l'entrée de la machine de Turing. La tête de lecture pointe sur le premier symbole, c'est-à-dire sur $\triangleright$.

A partir de la configuration initiale, la machine applique une transition selon δ, en changeant d'état, en écrivant un nouveau symbole, et en déplaçant sa tête de lecture. Elle applique ensuite une nouvelle transition et ainsi de suite. Les restrictions sur $\delta(q,\triangleright)$ permettent de faire en sorte que $\triangleright$ ne soit jamais remplacé par un autre symbole, et que la tête de lecture ne sorte pas à gauche du ruban.

Lorsque la tête arrive sur une case jamais lue, on considère que celle-ci contient le symbole blanc $\sqcup$, qui peut alors être récrit. Ainsi le contenu du ruban peut devenir arbitrairement long.

Comme δ est une application, la machine ne s'arrête à priori jamais. Cependant, elle s'arrête dans trois cas : en atteignant l'état h, l'état *yes* ou l'état *no*. Si elle atteint l'état *yes*, on dit que la machine *accepte*, si elle atteint l'état *no*, on dit qu'elle *rejette*. Si elle s'arête sur l'entrée x, on définit la sortie $M(x)$ de la machine comme suit. Si on arrive dans *yes*, $M(x) = yes$, si on arrive dans *no*, $M(x) = no$ et si on arrive dans h, $M(x)$ est le contenu du ruban au moment où la machine s'arête. Dans ce dernier cas, $M(x) = \triangleright y$ avec $y \in A^*$. Enfin, si la machine ne s'arrête pas sur x, on note $M(x) = \nearrow$

Pour une définition plus formelle d'un calcul d'une machine de Turing, on introduit la notion de configuration. Une *configuration* est un triplet (q,u,v), où $q \in Q$ et où uv est le contenu du ruban, c'est-à-dire que $uv \in \triangleright A^*$. Le triplet (q,u,v) représente une configuration de la machine dans laquelle l'état de contrôle est q, où le ruban contient uv, et où la tête de lecture pointe sur le premier symbole de v.

Considérons une configuration (q,u,v). On appelle a le premier symbole de $v = av''$ et l'on pose $\delta(q,a) = (q',a',D)$. Si $D = \rightarrow$, on pose $u' = ua'$ et $v' = v''$. Si $D = \leftarrow$, on pose $u = u'b$ et $v' = ba'v''$. Enfin, si $D = -$, on pose $u' = u$ et $v' = a'v''$. La configuration (q',u',v') est alors le successeur de (q,u,v), ce que l'on note $(q,u,v) \xrightarrow{M} (q',u',v')$. Enfin, on note $\xrightarrow{M^*}$ la clôture transitive de la relation $\xrightarrow{M}$.

Dans la suite, nous serons souvent amenés à simuler des machines de Turing. Pour cela on adoptera le codage suivant des configurations : une configuration (q,u,v) sera identifiée au mot uqv.

Le principal intérêt des machines de Turing est de caractériser des langages.

Définition 1.39 *Soit un langage L sur l'alphabet $A \setminus \{\sqcup\}$. Soit une machine de Turing M sur l'alphabet A, telle que pour toute entrée $x \in (A \setminus \{\sqcup\})^*$ si $x \in L$ alors $M(x) = yes$, et si $x \notin L$ alors $M(x) = no$. On dit alors que M* décide *L. Si L est décidé par une machine de Turing, on dit qu'il est* récursif *ou* décidable.

On dit que M accepte *un langage L lorsque pour toute entrée $x \in (A \setminus \{\sqcup\})^*$ si $x \in L$ alors $M(x) = yes$, et si $x \notin L$ alors $M(x) = \nearrow$. Si L est accepté par une machine de Turing, on dit qu'il est* récursivement énumérable *ou* demi-décidable.

Considérons enfin une application de $(A \setminus \{\sqcup\})^*$ dans A^* et une machine de Turing sur l'alphabet A. On dit que M calcule *f si pour toute entrée $x \in (A \setminus \{\sqcup\})^*$, $M(x) = f(x)$. Si une telle machine existe, f est une fonction* récursive *ou* calculable.

Les machines de Turing permettent donc de reconnaître des langages. Si l'on considère comme entrée le codage d'une instance d'un problème, on peut alors voir une machine de Turing comme un algorithme acceptant ou rejetant l'entrée. Ainsi, un problème (et un codage associé) sera dit *décidable* si le langage des codages des instances positives du problème est décidable.

On souhaite raffiner la notion précédente en considérant deux facteurs : le temps (c'est-à-dire le nombre de transitions) mis par la machine de Turing pour accepter, ainsi que l'espace utilisé par cette dernière. On définit alors les notions suivantes.

Définition 1.40 *Soit une fonction f de $\mathbb{N}$ dans lui même. Un langage L est décidé en temps $f(n)$ par une machine M, si M décide L, et si pour toute instance x, la machine accepte ou rejette x en $f(|x|)$ étapes de calcul au plus. Dans ce cas, on dit que $L \in \text{TIME}(f(n))$. On définit facilement la même notion pour les fonctions calculables.*

Définition 1.41 *Soit une fonction f de $\mathbb{N}$ dans lui même. Un langage L est décidé en espace $f(n)$ par une machine M si M décide L et si pour toute instance x, le nombre de symboles non blancs écrits sur le ruban lors du calcul de la machine sur x n'excède pas $f(|x|)$. Dans ce cas, on dit que $L \in \text{SPACE}(f(n))$. On définit facilement la même notion pour les fonctions calculables.*

Pour toute fonction f de $\mathbb{N}$ dans lui même, les classes de langages $\text{TIME}(f(n))$ et $\text{SPACE}(f(n))$ sont des *classes de complexité* en temps et en espace.

Comme dans le cas des automates finis (qui sont un cas particulier de machines de Turing qui lisent de gauche à droite sans récrire et qui s'arrêtent une fois le mot lu en entier), on définit les notions de machine de Turing non déterministes et alternantes. Nous nous contentons ici de donner des définitions informelles de ces deux notions.

Une *machine de Turing non déterministe* est un quadruplet $M = (Q,A,q_{in},\Delta)$, défini comme précédemment, sauf que maintenant Δ est à valeur dans $\mathcal{P}((Q \cup \{yes,no,h\}) \times A \times \{\leftarrow , \rightarrow ,-\})$. Ainsi, une configuration peut avoir plusieurs successeurs, et il n'y a donc plus un unique calcul sur une entrée donnée. Un langage $L \subseteq A^*$ est *accepté* par une machine de Turing non déterministe M si pour tout $x \in A^*$, $x \in L$ si et seulement s'il existe un calcul acceptant sur x

A nouveau, on définit les notions de temps et d'espace. On considère une fonction f de $\mathbb{N}$ dans lui-même. On dit qu'une machine de Turing non déterministe M décide un langage L en temps $f(n)$ si M décide L, et si pour toute entrée x, tout calcul de M sur x prend au plus $f(|x|)$ étapes. De même, on dit qu'une machine de Turing non déterministe M décide un langage L en espace $f(n)$ si M décide L, et si pour toute entrée x, le nombre de symboles écrits sur le ruban lors d'un calcul de la machine sur x n'excède pas $f(|x|)$. On note $\text{NTIME}(f(n))$ et $\text{NSPACE}(f(n))$ les classes de complexité associées.

Une *machine de Turing alternante* est un quadruplet $M = (Q,A,q_{in},\delta)$ défini comme précédemment, sauf que maintenant δ est à valeur dans $\mathcal{B}^+((Q \cup \{yes,no,h\}) \times A \times \{\leftarrow , \rightarrow ,-\})$. Ainsi, dans une configuration, la machine lance plusieurs calculs en parallèle correspondant à des transitions satisfaisant une formule booléenne positive. Un calcul est donc un arbre dont les branches sont des calculs au sens des

machines déterministes. Un calcul est acceptant si toutes les branches le composant sont des calculs acceptants. Un langage $L \subseteq A^*$ est *accepté* par une machine de Turing alternante M si pour tout $x \in A^*$, $x \in L$ si et seulement s'il existe un arbre de calcul acceptant sur x.

A nouveau, on définit les notions de temps et d'espace. On considère une fonction f de $\mathbb{N}$ dans lui-même. On dit qu'une machine de Turing alternante M décide un langage L en temps $f(n)$ si M décide L, et si pour toute entrée x, toute branche dans un calcul de M sur x prend au plus $f(|x|)$ étapes. De même, on dit qu'une machine de Turing alternante M décide un langage L en espace $f(n)$ si M décide L, et si, pour toute entrée x, le nombre de symboles écrits sur le ruban lors d'une branche du calcul de la machine sur x n'excède pas $f(|x|)$. On note $\text{ATIME}(f(n))$ et $\text{ASPACE}(f(n))$ les classes de complexité associées.

On donne maintenant quelques classes de complexité classiques :

- $\text{P} = \bigcup_{j>0} \text{TIME}(n^j)$.
- $\text{NP} = \bigcup_{j>0} \text{NTIME}(n^j)$.
- $\text{PSPACE} = \bigcup_{j>0} \text{SPACE}(n^j)$.
- $\text{NPSPACE} = \bigcup_{j>0} \text{NSPACE}(n^j)$.
- $\text{DEXPTIME} = \bigcup_{j>0} \text{TIME}(2^{n^j})$.
- $\text{EXPSPACE} = \bigcup_{j>0} \text{SPACE}(2^{n^j})$.
- k-DEXPTIME $= \bigcup_{j>0} \text{TIME}(tow(k,n^j))$, où la fonction $tow(k,n)$ est la tour d'exponentielle de hauteur k, c'est-à-dire que $tow(0,n) = n$ et $tow(k+1,n) = 2^{tow(k,n)}$
- $\text{ELEMENTARY} = \bigcup_{k>0}$ k-DEXPTIME est la classe des *problèmes élémentaires.*
- pour toute classe de complexité $\mathcal{C}$ on pose co$\mathcal{C} = \{L \mid \overline{L} \in \mathcal{C}\}$ la classe des langages dont le complémentaire est dans $\mathcal{C}$.

On donne quelques résultats classiques.

Proposition 1.8 *Pour toute fonction propre de complexité (voir [69] pour la définition précise) on a :*

- $\text{TIME}(f(n)) \subseteq \text{NTIME}(f(n)) \subseteq \text{SPACE}(f(n)) \subseteq \text{NSPACE}(f(n))$.
- $\text{NSPACE}(f(n)) \subseteq \bigcup_{j\geq 1} \text{TIME}(j^{\log n + f(n)})$.

Proposition 1.9 (Savitch) *Pour toute fonction propre de complexité $f(n) \geq \log n$, on a $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$. En particulier, on a l'égalité suivante :* $\text{NPSPACE} = \text{PSPACE}$.

Proposition 1.10 (Immerman-Szelepscényi) *Pour toute fonction propre de complexité $f(n) \geq \log n$, on a $\text{NSPACE}(f(n)) = \text{coNSPACE}(f(n))$.*

Enfin, on a le lien suivant entre les classes de complexité alternantes en espace et les classes de complexité déterministes en temps.

Proposition 1.11 *[25, 48, 24] Pour toute fonction propre de complexité $f(n) \geq \log n$, on a $\text{ASPACE}(f(n)) = \bigcup_{j\geq 1} \text{TIME}(j^{f(n)})$. En particulier, on a les égalités suivantes :*

- $\text{DEXPTIME} = \text{ASPACE}(n)$.

– k-DEXPTIME $= \bigcup_{j\geq 1}$ ASPACE$(tow(k-1,n^j))$, *pour tout* $k \geq 2$.

Etant donné un problème, on souhaitera déterminer s'il est décidable et le cas échéant dans quelle classe de complexité se trouve le langage des instances positives du problème. Cependant, on sera également intéressé par donner une borne supérieure. Pour cela, on introduit la notion de réduction polynomiale et de problème complet.

Définition 1.42 *Soient deux langages* L_1 *et* L_2*. On dit que* L_1 *se* réduit polynomialement *à* L_2 *s'il existe une fonction* f *calculable en temps polynomial telle que pour tout* x*,* $x \in L_1$ *si et seulement si* $f(x) \in L_2$*. En particulier, si* $L_2 \in \mathcal{C}$ *on a* $L_1 \in \mathcal{C}$ *pour toute classe* $\mathcal{C} =$ P,NP,*co*NP,PSPACE,DEXPTIME,*2*-DEXPTIME, ...

Soit une classe de complexité $\mathcal{C}$ *et un langage* L*. Le langage* L *est* $\mathcal{C}$-dur *si et seulement si tout problème dans* $\mathcal{C}$ *se réduit polynomialement à* L*. De plus, si* $L \in \mathcal{C}$*, il est dit* $\mathcal{C}$-complet*.*

On a alors le résultat suivant.

Proposition 1.12 *Soit une classe de complexité en temps* TIME$(f(n))$ *(respectivement* NTIME$(f(n))$*). Le langage des descriptions de machines de Turing déterministes (respectivement non déterministes) fonctionnant en temps* $f(n)$ *et acceptant l'entrée vide est* TIME$(f(n))$*-complet (respectivement* NTIME$(f(n))$*-complet).*

Soit une classe de complexité en espace SPACE$(f(n))$ *(respectivement* ASPACE$(f(n))$*). Le langage des descriptions des machines de Turing déterministes (respectivement alternantes) fonctionnant en espace* $f(n)$ *et acceptant l'entrée vide est* SPACE$(f(n))$*-complet (respectivement* ASPACE$(f(n))$*-complet).*

Par exemple, lorsque l'on souhaitera prouver qu'un problème est DEXPTIME-dur, il suffira de prouver qu'il est ASPACE(n)-dur, et pour cela, il suffira de montrer qu'il permet de simuler le comportement d'une machine de Turing alternante d'espace linéaire.

Enfin, on termine par le théorème de Cook.

Théorème 1.6 (Cook) *Le problème* SAT*, consistant à décider si une formule de la logique propositionnelle sous forme normale disjonctive avec trois littéraux par clause est satisfiable, est un problème* NP*-complet.*

Chapitre 2

Jeux à deux joueurs sur un graphe

Sommaire

Les jeux sont étudiés depuis longtemps dans trois domaines : en économie, en théorie descriptive des ensembles et en informatique théorique. Nous ne parlerons pas du premier, quant au second, il sera plus utilisé comme outil de preuve que réellement étudié. Notre centre d'intérêt concernera principalement le troisième, et tout particulièrement les jeux à deux joueurs sur des graphes. Les articles fondateurs sur le sujet sont entre autres [60, 15, 43, 14, 64, 30].

Pour une première approche des jeux, je ne saurais que trop recommander la lecture des excellents articles de W. Thomas [77] et de W. Zielonka [87]. Leur présentation est limpide et les références sont nombreuses.

Dans l'introduction, nous avons évoqué plusieurs motivations pour l'étude des jeux, en particulier le model-checking (vérification de modèle) et la synthèse de contrôleur. Le model-checking du μ-calcul et les jeux de parités sont très liés. Pour les articles *historiques* sur les structures finies, on consultera ceux d'E.A. Emerson, C.S. Jutla et A.P. Sistla [30, 31, 32], et pour les graphes de processus à pile ceux d'I. Walukiewicz [83, 84]. Enfin, on trouvera dans [6, 68, 85, 86] des présentations claires des liens entre le μ-calcul et les jeux. On se référera [5] pour la synthèse de

contrôleur et [8] pour les notions de permissivité. Pour une vision générale des liens entre les jeux et la vérification, on pourra consulter le tutoriel de W. Thomas [78] et pour les applications utilisant des jeux sur des graphes de processus à pile, on pourra consulter [23] et les références qui s'y trouvent.

Il y a eu ces dernières années plusieurs exposés invités sur les jeux dans des conférences. Leurs principal intérêt est qu'ils présentent des notions plus variées que celles exposées dans cette thèse et qu'ils donnent une bonne idée des problématiques liées aux jeux. On pourra consulter en particulier [28, 40, 85] et les références qui s'y trouvent.

Signalons également le recueil que constitue [41]. C'est une excellente référence pour qui s'intéresse aux jeux et la bibliographie y est extrêmement complète. Terminons enfin en signalant que la page web du projet GAMES, `http://www.games.rwth-aachen.de/index.html`, propose une très large bibliographie sur le sujet, qui offre un bon moyen de se tenir au courant de ce qui se passe dans la communauté scientifique sur le sujet.

Dans ce chapitre, on commence par donner les définitions classiques relatives aux jeux à deux joueurs sur un graphe, puis on explique ensuite comment définir un jeu sur un graphe de processus à pile, et on présente enfin les différentes conditions de gains qui seront étudiées dans cette thèse, avant d'introduire ensuite la notion de complexité borélienne et de terminer par quelques résultats classiques.

2.1 Définitions

2.1.1 Jeu, partie, condition de gain

Considérons les protagonistes, deux *joueurs*, que nous appellerons Eve et Adam. Soit $G = (V,E)$ un graphe quelconque.

On commence tout d'abord par considérer une partition des sommets $V = V_{\mathbf{E}} \cup V_{\mathbf{A}}$ entre les deux joueurs : les éléments de $V_{\mathbf{E}}$ seront les sommets (on parle aussi de positions) d'Eve et les éléments de $V_{\mathbf{A}}$ sont la propriété d'Adam.

Le triplet $\mathcal{G} = (G,V_{\mathbf{E}},V_{\mathbf{A}})$ est un *graphe de jeu*. On représente un graphe de jeu comme un graphe, à la différence que les sommets d'Eve sont des cercles et ceux d'Adam des carrés. La figure 2.1 est la représentation graphique d'un graphe de jeu $\mathcal{G} = (G,V_{\mathbf{E}},V_{\mathbf{A}})$ où $V_{\mathbf{E}} = \{2,4,7,8\}$ et $V_{\mathbf{A}} = \{1,3,5,6\}$.

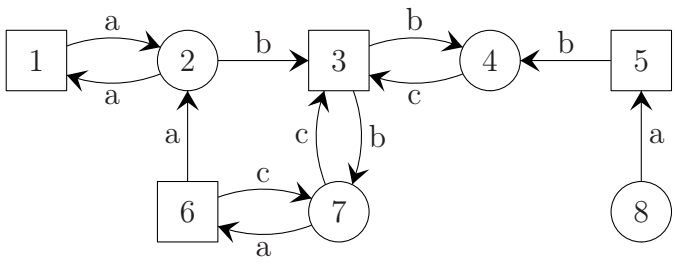


FIG. 2.1 – *Exemple de graphe de jeu*

Une *condition de gain* sur $\mathcal{G}$ est un sous-ensemble Ω de $V \times E^\infty$.

Enfin, un *jeu à deux joueurs* sur un graphe de jeu $\mathcal{G}$ est un couple $\mathbb{G} = (\mathcal{G},\Omega)$, où $\mathcal{G}$ est un graphe de jeu et Ω une condition de gain sur $\mathcal{G}$.

Une *partie* dans un jeu $\mathbb{G} = (\mathcal{G},\Omega)$ au départ d'une position initiale v_0 se déroule de la façon suivante. Le joueur à qui appartient la position initiale v_0 choisit, si cela est possible, un arc e_1 d'origine v_0. S'il n'existe pas d'arc d'origine v_0, la partie se termine. Sinon, le joueur à qui appartient l'extrémité v_1 de e_1 choisit un arc e_2 d'origine v_1 si cela est possible et ainsi de suite. La partie est donc un chemin $(v_0,e_1e_2\cdots)$, fini ou infini, dans G, et peut être vue comme un élément de $V \times E^\infty$. Par *partie partielle*, on désignera un préfixe d'une partie, c'est-à-dire un couple $(v_0,e_1\cdots e_n)$ défini comme précédemment, à la différence près que le chemin $(v_0,e_1\cdots e_n)$ peut être éventuellement prolongeable. Une partie partielle est donc un élément de $V \times E^*$. Autant que faire se peut, on réservera les lettres Λ et λ pour désigner des parties ou des parties partielles.

Reprenons le graphe de jeu donné dans la figure 2.1. Le couple $[1,(1,a,2)(2,b,3)(3,b,7)(7,a,6)(6,a,2)((2,a,1)(1,a,2))^\omega]$ est une partie dans le graphe de jeu précédent.

Eve remporte une partie si et seulement si elle appartient à Ω, sinon, c'est Adam qui gagne. Une convention classique, que nous adopterons fréquemment par la suite, consiste à déclarer comme perdant d'une partie finie (c'est-à-dire une partie qui aboutit dans un cul-de-sac) le joueur qui ne peut bouger. Dans ce cadre, la condition de gain ne traitera que des parties infinies et sera alors un sous-ensemble de $V \times E^\omega$.

Dans la suite, nous considérerons principalement deux types de conditions de gains : les conditions *internes* qui ne considèrent que la suite des sommets visités au cours d'une partie et les conditions *externes* qui ne considèrent que la suite des étiquettes des arcs traversés au cours d'une partie.

Définition 2.1 *Une* condition de gain interne *sur un graphe de jeu $\mathcal{G}$ de sommets V est un sous-ensemble Ω de V^∞. Une partie $\Lambda = (v_0,e_0e_1e_2\cdots)$ est remportée par Eve si et seulement si $v_0v_1v_2\cdots$ appartient à Ω où v_{i+1} désigne l'extrémité de e_i pour tout $0 \leq i < |\Lambda|$.*

Définition 2.2 *Une* condition de gain externe *sur un graphe de jeu A-étiqueté $\mathcal{G}$ est un sous-ensemble Ω de A^∞. Une partie $\Lambda = (v_0,e_0e_1e_2\cdots)$ est remportée par Eve si et seulement si $\mathrm{Lab}(\Lambda) = a_0a_1a_2\cdots$ appartient à Ω où a_i désigne l'étiquette de e_i pour tout $0 \leq i < |\Lambda|$.*

Etant donnée une condition de gain Ω, on considère la *condition duale* $\overline{\Omega}$ définie comme $\overline{\Omega} = (V \times E^\infty) \setminus \Omega$. Dans le cas particulier où l'on considère que les parties finies sont perdues par le joueur bloqué, on définit $\overline{\Omega} = (V \times E^\omega) \setminus \Omega$. Ainsi, une partie est remportée par Adam dans le jeu $\mathbb{G} = (\mathcal{G},\Omega)$ si et seulement si elle appartient à $\overline{\Omega}$. Il est clair que la condition duale d'une condition interne est une condition interne, et que la condition duale d'une condition externe est une condition externe.

Donnons maintenant quelques exemples de conditions de gains. On commence par les conditions d'accessibilité et de sûreté

Définition 2.3 *Soit un sous-ensemble $F \subseteq V$ de sommets, appelés* sommets finaux.

*La condition interne V^*FV^∞ est une condition dite d'*accessibilité*. Ainsi, Eve remporte une partie si un état final est visité au cours de celle-ci.*

La condition duale $(V \setminus F)^\omega$ d'une condition d'accessibilité est une condition dite de sûreté*. On parle alors pour F de sommets* interdits*. Ainsi, Eve remporte une partie si on ne visite pas d'état interdit au cours de celle-ci.*

Pour les condition de gains suivantes, on adopte la convention qu'une partie finie est perdue par le joueur qui ne peut bouger. On définit alors les conditions de Büchi, de co-Büchi, de parité et de Muller.

Définition 2.4 *Soit un sous-ensemble $F \subseteq V$ de sommets, appelés* sommets finaux*. La condition interne $(V^*F)^\omega$ est une condition dite de* Büchi*. Ainsi, Eve remporte une partie si on visite infiniment souvent des états finaux au cours de celle-ci.*

La condition duale $\bigcup_{i\geq 0}(V^*F)^i(V \setminus F)^\omega$ d'une condition de Büchi est une condition dite de co-Büchi*. On parle alors pour F de sommets* interdits*. Ainsi, Eve remporte une partie si on ne visite qu'un nombre fini d'états interdits au cours de celle-ci.*

Définition 2.5 *Soit un sous-ensemble fini $C \subseteq \mathbb{N}$ d'entiers positifs appelés* couleurs *et soit une application ρ de V dans C, appelée* fonction de coloriage*. La condition interne $\{v_0v_1v_2\cdots \in V^\omega \mid \liminf(\rho(v_i))_{i\geq 0}$ est paire$\}$ est une condition dite de* parité*. Ainsi, Eve remporte une partie si la plus petite couleur infiniment souvent visitée est paire.*

La condition duale $\{v_0v_1v_2\cdots \in V^\omega \mid \liminf(\rho(v_i))_{i\geq 0}$ est impaire$\}$ est une condition dite de co-parité*. Ainsi, Eve remporte une partie si la plus petite couleur infiniment souvent visitée est impaire.*

Remarque 2.1 Considérons une condition Ω de co-parité sur un graphe de jeu coloré par une fonction de coloriage ρ. On considère alors la fonction de coloriage ρ' définie par $\rho'(v) = \rho(v) + 1$. Il est facile de voir que la condition de parité Ω' sur le graphe de jeu précédent coloré par la fonction ρ', est telle que $\Omega = \Omega'$.

Définition 2.6 *Soit un sous-ensemble fini $C \subseteq \mathbb{N}$ d'entiers positifs appelé* couleurs*, et soit une application ρ de V dans C, appelée* fonction de coloriage*. Soit une collection $\mathcal{F}$ de sous-ensembles de C, appelés* ensembles finaux*. La condition interne $\{v_0v_1v_2\cdots \in V^\omega \mid \{c \mid \exists^\infty i,\ \rho(v_i) = c\} \in \mathcal{F}\}$ est une condition dite de* Muller*. Ainsi, Eve remporte une partie si l'ensemble des couleurs infiniment souvent visitées est dans $\mathcal{F}$.*

La condition duale d'une condition de Muller d'ensembles finaux $\mathcal{F}$ est également une condition de Muller avec la même fonction de coloriage et avec $\mathcal{P}(C) \setminus \mathcal{F}$ pour ensembles finaux.

Donnons enfin deux exemples de conditions externes.

Définition 2.7 *Considérons un langage ω-régulier L sur un alphabet d'étiquetage A. Soit un graphe de jeu A-étiqueté. Le langage $\Omega = L$ est une condition dite ω-*régulière*. La condition duale d'une condition ω-régulière est également ω-régulière, par clôture par complémentation des langages réguliers.*

Définition 2.8 *Considérons un langage* ω-VPL L *sur un alphabet d'actions* A*. Soit un graphe de jeu* A*-étiqueté. Le langage* $\Omega = L$ *est une condition dite* ω-VPL*. La condition duale d'une condition* ω-VPL *est également* ω-VPL*, par clôture par complémentation des langages* ω*-VPL[4].*

2.1.2 Stratégies et positions gagnantes

Dans toute ce paragraphe, on se donne un graphe de jeu $\mathcal{G} = (G,V_{\mathbf{E}},V_{\mathbf{A}})$, où $G = (V,E)$.

Stratégies

Qui dit jeu dit envie de gagner, et donc stratégie. Une stratégie consiste à considérer les coups déjà joués ainsi que la position courante pour décider quel sommet atteindre. Plus formellement, une *stratégie* pour Eve est une fonction $\varphi : V \times E^* \to E$ telle que, pour toute partie partielle λ se terminant en $v \in V_{\mathbf{E}}$, $\varphi(\lambda)$ a pour origine v. De même, on définit les stratégies pour Adam comme des fonctions $\psi : V \times E^* \to E$ telles que, pour toute partie partielle λ se terminant en $v \in V_{\mathbf{A}}$, $\psi(\lambda)$ a pour origine v.

Il existe d'autres définitions plus générales pour les stratégies. En particulier, on peut les définir non plus comme des fonctions à valeur dans E, mais comme des applications à valeur dans $\mathcal{P}(E)$. Une *stratégie non déterministe* pour Eve est une application $\varphi : V \times E^* \to \mathcal{P}(E)$ telle que, pour toute partie partielle λ se terminant en $v \in V_{\mathbf{E}}$, les éléments de $\varphi(\lambda)$ ont pour origine v. De même, on définit les stratégies pour Adam comme des applications $\psi : V \times E^* \to E$ telles que, pour toute partie partielle λ se terminant en $v \in V_{\mathbf{A}}$, les éléments de $\psi(\lambda)$ ont pour origine v.

Les stratégies non déterministes généralisent bien les stratégies déterministes. En effet, étant donnée une stratégie déterministe φ, on définit un stratégie φ' équivalente en posant $\varphi'(\lambda) = \emptyset$ pour tout λ si φ n'est pas définie en λ, et $\varphi'(\lambda) = \{\varphi(\lambda)\}$ sinon. Pour la suite des définitions, on se place dans le cadre des stratégies non déterministes.

Etant donnée une stratégie φ pour Eve, cette dernière peut la *respecter* en choisissant toujours de prendre un arc donné par φ sur le préfixe de partie déjà joué. Plus formellement, on dira qu'Eve *respecte* φ au cours d'une partie $\lambda = (v,e_1e_2\cdots)$ si, pour tout $0 \leq i < |\lambda|$, $e_{i+1} \in \varphi((v,e_1\cdots e_i))$. De même, on définit le fait qu'Adam respecte une stratégie ψ.

Les stratégies intéressantes sont celles que l'on peut suivre tout au long d'une partie. Plus précisément, on dira qu'une stratégie φ pour Eve est *praticable* depuis un sommet v si, pour toute partie partielle λ d'origine v où Eve respecte φ, et se terminant par un arc d'extrémité dans $V_{\mathbf{E}}$, $\varphi(\lambda)$ est non vide. De même, on définit les stratégies praticables pour Adam.

Remarque 2.2 Evoquons maintenant le cas particulier des graphes non étiquetés. Dans la remarque 1.1, nous avions noté que l'on peut représenter les chemins par des mots sur l'alphabet des sommets. Dans ce formalisme, une stratégie déterministe pour Eve est alors une application φ de V^∞ dans $\mathcal{P}(V)$ telle que, pour toute partie

partielle λ se terminant dans un sommet $v \in V_{\mathbf{E}}$ et pour tout $v' \in \varphi(\lambda)$, $(v,v') \in E$. De façon symétrique, on définit les stratégies pour Adam.

Etant donnée une stratégie φ, Eve respecte φ lors d'une partie $\lambda = v_1 v_2 v_3 \cdots$ si, pour tout $1 \leq i < |\lambda|$, $v_i \in V_{\mathbf{E}} \Rightarrow v_{i+1} \in \varphi(v_1 v_2 \cdots v_i)$.

Considérons maintenant un jeu $\mathbb{G} = (\mathcal{G},\Omega)$ sur $\mathcal{G}$, une position initiale v_0 et une stratégie φ pour Eve. On dira que φ est une *stratégie gagnante pour Eve dans* $\mathbb{G}$ *depuis* v_0 si toute partie au départ de v_0 où Eve respecte φ, est remportée par Eve. On définit de même les stratégies gagnantes pour Adam. Le jeu $\mathbb{G}$ est *déterminé* si, pour toute position initiale, l'un des deux joueurs possède une stratégie gagnante depuis cette position dans $\mathbb{G}$.

Remarque 2.3 Etant donné un jeu, l'un des deux joueurs au plus a une stratégie gagnante. En effet, si les deux joueurs possédaient une stratégie gagnante, la partie obtenue quand ces derniers respectent leurs stratégies gagnantes respectives serait à la fois dans et hors de Ω.

Remarque 2.4 Etant donné une stratégie φ gagnante pour Eve depuis un sommet v dans un jeu $\mathbb{G}$, il est facile de voir qu'Eve possède une stratégie gagnante déterministe depuis v. En effet, il suffit de considérer la stratégie φ' définie pour tout $\lambda \in V \times E^\infty$ tel que $\varphi(\lambda)$ est non vide, par $\varphi'(\lambda) = e_\lambda$, où e_λ est un élément quelconque de $\varphi(\lambda)$. On vérifie alors facilement que φ' est gagnante pour Eve.

Positions gagnantes

Une position v dans un jeu $\mathbb{G}$ est une *position gagnante* pour Eve si celle-ci possède une stratégie gagnante dans $\mathbb{G}$ depuis v. L'ensemble des positions gagnantes pour Eve sera généralement noté $W_{\mathbf{E}}$. On définit de même l'ensemble $W_{\mathbf{A}}$ des positions gagnantes pour Adam.

Stratégies et représentations

On se fixe maintenant un graphe de jeu $\mathcal{G} = ((V,E),V_{\mathbf{E}},V_{\mathbf{A}})$, et un jeu $\mathbb{G} = (\mathcal{G},\Omega)$ sur $\mathcal{G}$. Dans ce paragraphe, nous discutons des différents types de stratégies possibles et de leurs représentations respectives.

Dans le cas général, une stratégie est un objet qui n'est pas finiment descriptible puisqu'il s'agit d'une application de $V \times E^\infty$ dans $\mathcal{P}(E)$. Cependant, afin de manipuler ces dernières dans certaines preuves, on utilisera parfois une représentation par un arbre a priori infini, que l'on décrit ci-dessous.

Soit un sommet $v \in V$ et soit φ une stratégie praticable pour Eve depuis v. On associe à φ sa représentation par un arbre $\mathcal{T}_\varphi$, appelé *arbre de stratégie* et défini récursivement de la façon suivante :

- les nœuds de $\mathcal{T}_\varphi$ sont étiquetés par des éléments dans E, à l'exception de la racine qui est étiquetée par v.
- étant donné un nœud n de $\mathcal{T}_\varphi$ d'étiquette e, on appelle u l'extrémité de e si $e \in E$ et $u = v$ sinon. Si $u \in V_{\mathbf{A}}$, alors n a autant de fils qu'il y a d'arcs d'origine u dans $\mathcal{G}$, et chaque fils est étiqueté par l'un de ces arcs (et réciproquement).

Si $u \in V_{\mathbf{E}}$ alors on considère la suite $v,e_1,e_2,\ldots,e$ des étiquettes des nœud allant de la racine à n (inclus), et l'on note $\lambda = (v,e_1e_2\cdots e)$ la partie partielle qu'elles définissent. Le nœud n a alors pour fils des nœuds étiquetés par les éléments de $\varphi(\lambda)$ (il y a un unique fils si φ est déterministe et plusieurs si φ est non déterministe).

Ainsi, les branches de l'arbre $\mathcal{T}_\varphi$ décrivent l'ensemble de toutes les parties possibles lorsque Eve suit sa stratégie φ au départ de v. On a alors facilement le résultat suivant.

Proposition 2.1 *Une stratégie φ praticable pour Eve depuis un sommet v est gagnante pour Eve dans le jeu $\mathbb{G}$ au départ de v si et seulement si, pour toute branche b dans $\mathcal{T}_\varphi$, la partie $\lambda = (v,e_1e_2\cdots)$ est dans Ω, où la suite $v,e_1,e_2,\ldots$ est la suite des étiquettes des nœuds de la branche b lue depuis la racine.*

Evoquons maintenant le cas des stratégies dites *sans mémoire* ou *positionnelles*. Une stratégie est sans mémoire si le choix du coup à jouer ne dépend pas du passé de la partie mais uniquement de la position courante.

Définition 2.9 *Une stratégie φ est* sans mémoire *(ou* positionnelle*) si pour toutes parties partielles λ et λ' de même extrémité, $\varphi(\lambda) = \varphi(\lambda')$. Ainsi, les stratégies sans mémoire pour Eve (respectivement pour Adam) sont exactement l'ensemble des applications φ de $V_{\mathbf{E}}$ (respectivement de $V_{\mathbf{A}}$) dans $\mathcal{P}(E)$ telles que pour tout $v \in V$, les éléments de $\varphi(v)$ ont pour origine v. Dans le cas des graphes non étiquetés, ce sont les fonctions de $V_{\mathbf{E}}$ (respectivement de $V_{\mathbf{A}}$) dans $\mathcal{P}(V)$, telles que pour tout $v \in V_{\mathbf{E}}$ et tout $v' \in \varphi(v)$, $(v,v') \in E$.*

Dans le cas où le graphe de jeu est fini, une stratégie sans mémoire est donc représentable de façon finie. Dans le cas où le graphe est infini, ce n'est pas le cas en général, puisque le graphe lui-même peut ne pas être finiment représentable. C'est pourquoi on préférera parler de stratégie sans mémoire pour les jeux sur des graphes finis, et de stratégie positionnelle pour les jeux sur des graphes infinis.

Entre les deux extrêmes que sont les stratégies générales et les stratégies positionnelles, le concept de *stratégies avec mémoire* permet de définir les deux notions précédentes mais aussi de les raffiner.

Définition 2.10 *Soit M un ensemble que l'on appelle* mémoire*. Une* stratégie à mémoire *M pour Eve est un triplet (m_0,φ,up) formé d'un élément $m_0 \in M$ appelé* mémoire initiale*, d'une application φ de $V \times M$ dans $\mathcal{P}(E)$ telle que pour tout couple (v,m), tout élément de $\varphi(v,m)$ est d'origine v, et d'une application up de $M \times E$ dans M appelée* application de mise à jour*. Eve respecte φ au cours d'une partie $\lambda = (v_0,e_1e_2,\cdots)$ si pour tout $i < |\lambda|$, $e_{i+1} \in \varphi(v_i,m_i)$, où v_i est l'extrémité de e_i si $i \geq 1$ et où l'on pose $m_i = up(m_{i-1},e_i)$ pour tout $i \geq 1$. En d'autres termes, quand Eve doit jouer, elle regarde la valeur de la mémoire ainsi que le sommet courant pour déterminer quel arc emprunter. Après chaque coup (effectué par Eve ou par Adam), Eve met à jour sa mémoire à l'aide de la fonction up en considérant l'ancienne valeur et le dernier coup joué. La stratégie φ est gagnante si toutes les parties où Eve respecte φ sont gagnantes. On définit également le caractère praticable d'une stratégie à mémoire. Il est alors facile de définir les mêmes notions pour Adam.*

Lorsque M est réduit à un seul élément (et n'est donc pas utile), on retrouve les stratégies sans mémoire (d'où leur nom).

Soit une stratégie φ générale et soit v une position initiale. Considérons la stratégie à mémoire M, (m_0,ψ,up) où l'on pose $M = V \times E^\infty$, $m_0 = (v,\varepsilon)$, $\psi(v,m) = \varphi(m)$ et $up((v,e_1 \cdots e_n),e_{n+1}) = (v,e_1 \cdots e_n e_{n+1})$. La stratégie ci-dessus stocke donc dans sa mémoire le préfixe de partie déjà joué. Dès lors il est évident que l'ensemble des parties où Eve respecte la stratégie (m_0,ψ,up) est exactement l'ensemble des parties où Eve respecte φ.

Un cas particulièrement intéressant est celui des *stratégies à mémoire finie*, c'est-à-dire des stratégies à mémoire M où M est fini. Lorsque le graphe de jeu est fini, une telle stratégie est réalisée par un transducteur (c'est-à-dire un automate fini avec sortie).

Terminons enfin en définissant une variante des stratégies sans mémoire, les *stratégies persistantes* [57]. Dans une stratégie sans mémoire, on joue toujours de la même façon dans un même sommet et le coup choisi est fixé par avance. Une généralisation est de considérer que l'on joue toujours le même coup dans un même sommet mais que ce choix n'est fixé que lors de la première visite dans le sommet.

Définition 2.11 *Une stratégie déterministe φ est* persistante *si pour toutes parties λ et λ' de même extrémité et telle que λ' prolonge λ, $\varphi(\lambda) = \varphi(\lambda')$.*

Il est facile de voir que, même dans le cas des jeux sur des graphes finis, de telles stratégies peuvent ne pas avoir de représentation finie.

2.2 Jeux sur des graphes de processus à pile

Dans le paragraphe 1.3.1 nous avons montré comment construire, à partir d'un processus à pile, un graphe infini. On explique maintenant comment définir, à partir d'un graphe de processus à pile, un graphe de jeu. On explique ensuite comment définir les conditions de gain classiques et l'on introduit également de nouvelles conditions de gains.

2.2.1 Définition du graphe de jeu et représentation

Définition 2.12 *Soit un processus à pile $\mathcal{P} = \langle Q,A,\Gamma,\bot,\Delta\rangle$ et soit $G = (V,E)$ le graphe engendré par $\mathcal{P}$. Soit une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de Q entre Eve et Adam. La partition précédente induit une partition $V_{\mathbf{E}} \cup V_{\mathbf{A}}$ de V définie par $V_{\mathbf{E}} = Q_{\mathbf{E}} \times (\Gamma \setminus \{\bot\})^*\bot$ et $V_{\mathbf{A}} = Q_{\mathbf{A}} \times (\Gamma \setminus \{\bot\})^*\bot$: les configurations d'Eve sont celles dont l'état de contrôle est dans $Q_{\mathbf{E}}$. Le graphe de jeu $\mathcal{G} = (G,V_{\mathbf{E}},V_{\mathbf{A}})$ est appelé* graphe de jeu sur le processus à pile $\mathcal{P}$ induit par $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$.

Considérons un processus à pile $\mathcal{P} = \langle Q,A,\Gamma,\bot,\Delta\rangle$, une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de Q, et appelons $\mathcal{G}$ le graphe de jeu associé. Etant donnée une configuration $v = (q,\sigma)$, on appelle *hauteur* de v, que l'on note $|v|$, la taille $|\sigma|$ du contenu de pile dans v.

Les hauteurs de pile vont jouer un rôle crucial dans la résolution des jeux sur des graphes de processus à pile. On s'intéressera essentiellement à savoir si dans une configuration de hauteur donnée, on revient un jour dans une configuration de même

hauteur. Pour illustrer les raisonnements, on adoptera la représentation d'une partie par un graphique défini comme suit : l'axe des abscisses représente le temps tandis que l'axe des ordonnées représente la hauteur de pile. Un tel graphique permet donc de lire l'évolution de la hauteur de pile au cours de la partie. La figure 2.2 donne un exemple d'une telle représentation : on commence par empiler puis dépiler (on parle de bosse). Ensuite on empile deux symboles puis un troisième que l'on dépile. Ensuite on empile un symbole puis un second que l'on dépile...

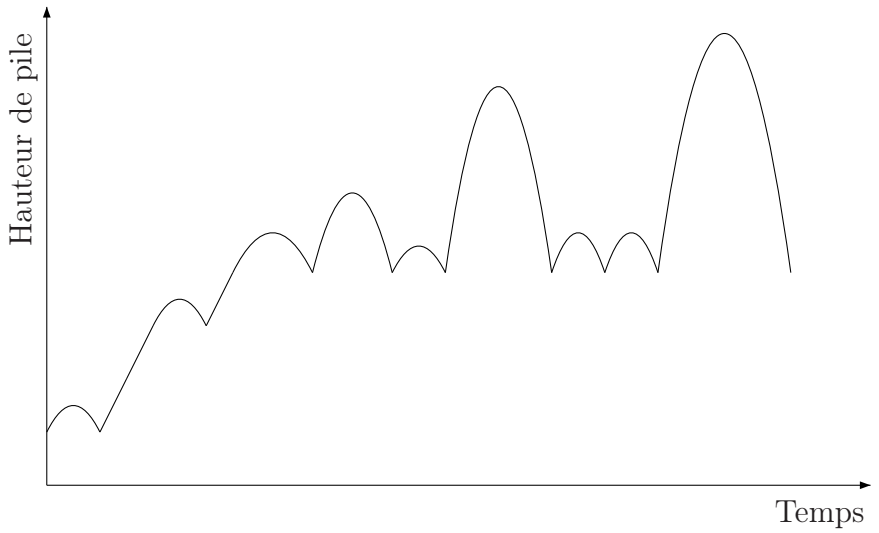


FIG. 2.2 – *Représentation graphique d'une partie*

2.2.2 Conditions d'accessibilité, de Büchi et de parité

Dans la construction du graphe de jeu, on a utilisé les états de contrôle pour partager les configurations entre les deux joueurs. La même idée nous permet de définir des configurations finales et des fonctions de coloriage. Les jeux de Büchi et de parité ne considérant pas l'étiquetage des arcs, on considère ici des graphes de jeu engendrés par des processus à pile non étiqueté.

Définition 2.13 (Jeu d'accessibilité, de Büchi) *Soit un graphe de jeu engendré par un processus à pile* $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ *et une partition* $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ *de* Q*. Soit* $\mathcal{G}$ *le graphe de jeu engendré par* $\mathcal{P}$ *et la partition précédente. Soit un sous-ensemble* F *de* Q*. L'ensemble* F *induit alors un ensemble* $V_F = F \times (\Gamma \setminus \{\bot\})^*\bot$ *de configurations finales.*

Soit Ω_{Acc} *la condition d'accessibilité associée à* V_F *sur* $\mathcal{G}$*. Le jeu* $\mathbb{G} = (\mathcal{G},\Omega_{Acc})$ *est qualifié de* jeu d'accessibilité sur un graphe de processus à pile.

Soit Ω_{Buc} *la condition de Büchi associée à* V_F *sur* $\mathcal{G}$*. Le jeu* $\mathbb{G} = (\mathcal{G},\Omega_{Buc})$ *est qualifié de* jeu de Büchi sur un graphe de processus à pile.

Définition 2.14 (Jeu de Parité) *Soit un graphe de jeu engendré par un processus à pile* $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ *et une partition* $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ *de* Q*. Soit* $\mathcal{G} = ((V,E),V_{\mathbf{E}},V_{\mathbf{A}})$

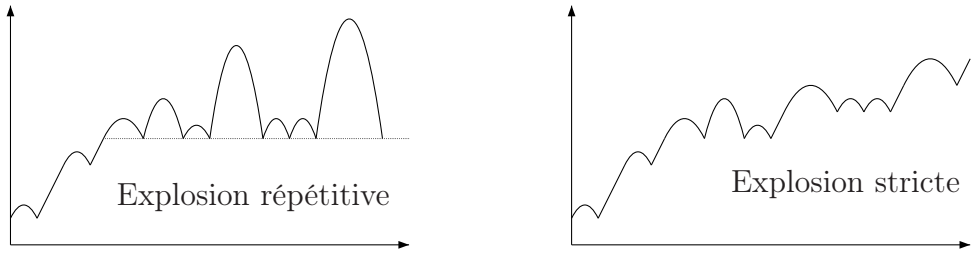


FIG. 2.3 – *Les deux types de parties gagnantes pour l'explosion*

le graphe de jeu engendré par $\mathcal{P}$ et la partition précédente. Soit une fonction de coloriage ρ de Q dans un ensemble de couleur C. On étend ρ en une fonction de V dans C en posant $\rho'((q,\sigma)) = \rho(q)$ pour tout état q et tout contenu de pile σ. On appelle Ω la condition de parité associée à ρ' sur $\mathcal{G}$. Le jeu $\mathbb{G} = (\mathcal{G},\Omega)$ est qualifié de jeu de parité sur un graphe de processus à pile.

2.2.3 Conditions de gain sur la hauteur de pile

Les processus à pile permettent de modéliser des programmes autorisant des appels récursifs, stockés dans la pile. Dans ce formalisme, la hauteur de pile représente le nombre d'appels récursifs en cours. Il est alors naturel de se demander si la pile est bornée ou pas au cours d'une partie, ce qui exprime le fait que le nombre d'appels récursifs soit borné ou non.

On considère un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ et une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de Q. On appelle $\mathcal{G} = ((V,E),V_{\mathbf{E}},V_{\mathbf{A}})$ le graphe de jeu engendré par $\mathcal{P}$ et la partition précédente. Pour tout entier $n \geq 1$, on note V_n l'ensemble des configurations de $\mathcal{G}$ de hauteur de pile égale à n, par $V_{\leq n}$ l'ensemble des configurations de $\mathcal{G}$ de hauteur de pile inférieure ou égale à n et par $V_{\geq n}$ l'ensemble des configurations de $\mathcal{G}$ de hauteur de pile supérieure ou égale à n.

La condition de gain $\Omega_{Exp} = \bigcap_{n\geq 1}[(V^*V_n)V^\omega]$ est qualifiée de *condition d'explosion*. Le jeu $\mathbb{G} = (\mathcal{G},\Omega_{Exp})$ dans lequel on adopte la convention qu'une partie finie est perdue par le joueur qui ne peut bouger est qualifié de *jeu d'explosion*.

La condition duale $\Omega_{Bor} = \bigcup_{n\geq 1} V^\omega_{\leq n}$ de la condition d'explosion est qualifiée de *condition de bornage*.

Une partie est remportée par Eve dans un jeu d'explosion si et seulement si la pile est non bornée. On peut distinguer deux types possibles de partie gagnantes pour Eve, selon qu'un niveau de pile est infiniment souvent visité ou non. Ces deux cas sont représentés dans la figure 2.3. Une partie est remportée par Eve pour la condition de bornage si et seulement si la hauteur de pile est bornée.

On définit une variante de la condition d'explosion, la condition d'explosion stricte (dont l'interprétation graphique est donnée dans la figure 2.3).

La condition de gain $\Omega_{ExpSt} = \bigcap_{n\geq 1} V^*V^\omega_{\geq n}$ est qualifiée de *condition d'explosion stricte*. Le jeu $\mathbb{G} = (\mathcal{G},\Omega_{ExpSt})$, dans lequel on adopte la convention qu'une partie finie est perdue par le joueur qui ne peut bouger, est qualifié de *jeu d'explosion stricte*.

La condition duale Ω_{Rep} de la condition d'explosion stricte, que l'on appelle *condition de répétition* a été étudiée dans [21] et est donnée par $\bigcup_{n\geq 1}\bigcap_{m\geq 0}[(V^m V^* V_n)V^\omega]$.

Une partie est remportée par Eve dans un jeu d'explosion stricte si et seulement si toute hauteur de pile est un jour quittée pour toujours. Une partie est remportée par Adam si et seulement si un niveau de pile est infiniment souvent visité, ou de façon équivalente si et seulement si une configuration est infiniment souvent visitée.

Pour une partie $\Lambda = (p_0,\sigma_0)(p_1,\sigma_1)(p_2,\sigma_2)\cdots$ remportée par Eve dans un jeu d'explosion stricte, on définit la *limite de pile* dans Λ, notée $\mathrm{StLim}(\Lambda)$, et définie par $\mathrm{StLim}(\Lambda) = \lim(\sigma_i)_{i\geq 0}$.

Dans le chapitre 6, on considérera des conditions demandant que la pile explose strictement et que sa limite appartienne à un langage donné de mots infinis.

2.2.4 Condition de parité en escalier

On définit maintenant une nouvelle condition que l'on qualifie de *parité en escalier*. Informellement, cette condition est la même que la condition de parité, sauf que l'on évalue la condition de parité, non pas sur la suite de toutes les couleurs, mais sur la sous-suite des couleurs associée aux configurations de hauteur de pile minimale par rapport au futur de la partie.

Plus précisément, on se donne un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ et une partition $Q_{\mathbf{E}}\cup Q_{\mathbf{A}}$ de Q. Soit $\mathcal{G} = ((V,E),V_{\mathbf{E}},V_{\mathbf{A}})$ le graphe de jeu engendré par $\mathcal{P}$ et la partition précédente. Soit une fonction de coloriage ρ de Q dans un ensemble de couleur C que l'on étend en une application ρ' de V dans C en posant $\rho'((q,\sigma)) = \rho(q)$ pour tout état q et tout contenu de pile σ.

A toute partie infinie $\Lambda = v_0v_1v_2$ dans $\mathcal{G}$, on associe l'ensemble infini $Steps_\Lambda$, défini par $Steps_\Lambda = \{n \in \mathbb{N} \mid \forall m \geq n, |v_m| \geq |v_n|\}$.

La figure 2.4 illustre une partie Λ dans laquelle les positions dont l'indice est dans $Steps_\Lambda$ sont marquées par un rond.

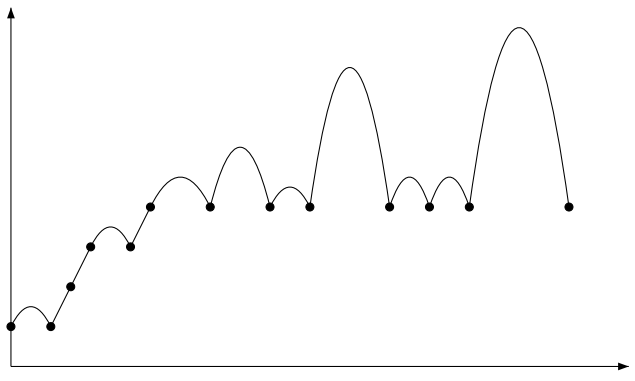


FIG. 2.4 – $Steps_\Lambda$

Enfin, la condition de parité en escalier est la condition interne

$$\Omega = \{\Lambda = v_0v_1v_2\cdots \mid \min\{c \mid \exists^\infty i \in Steps_\Lambda \text{ t.q. } \rho'(v_i) = c\} \text{ est pair}\}$$

2.2.5 Stratégie à pile

Dans le cas des jeux sur des graphes de processus à pile, on s'intéressera à une famille particulière de stratégies à mémoire, les *stratégies à pile*. Informellement, ces stratégies sont exactement celles réalisables par un automate à pile avec sortie lisant les coups joués au cours de la partie et sortant les coups à jouer.

Définition 2.15 *Soit un jeu $\mathbb{G}$ sur un graphe de jeu engendré par un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$. Une* stratégie à pile *dans $\mathbb{G}$ est une stratégie à mémoire (m_0,φ,up) sur une mémoire $M = S\times(\Pi\setminus\{\bot\})^*\bot$, où S est un ensemble d'états fini, et Π est un alphabet de pile de stratégie qui est fini. De plus, pour toute configuration $v = (q,\sigma)$, tout état s, et tout contenu de pile de stratégie ν, $\varphi(v,(s,\nu))$, ne dépend que q, du sommet de σ, de s et du sommet de ν. Par ailleurs, $\varphi(v,(s,\nu))$ est à valeur dans $\{push(q,\gamma),pop(q),skip(q) \mid q \in Q,\gamma \in \Gamma\}$, ce qui permet bien de déterminer de façon unique la configuration à atteindre. Enfin, on demande que la fonction de mise à jour de la stratégie up soit une application de $(S \times \Pi) \times \{push(q,\gamma),pop(q),skip(q) \mid q \in Q,\gamma \in \Gamma\}$ dans $\{push(s,\pi),pop(s),skip(s) \mid s \in S, \pi \in \Pi\}$. La mémoire est alors mise à jour comme les configurations d'un automate à pile, par exemple si $up((s,\pi),pop(q)) = push(s',\pi')$, on change l'état de stratégie en s' et l'on empile π' dans la pile de stratégie.*

L'intérêt principal de telles stratégies réside dans le fait qu'elles sont finiment descriptibles. Une telle propriété est d'autant plus intéressante que les graphes de jeu issus de processus à pile sont eux-même finiment représentables.

Remarque 2.5 Dans la définition précédente, on considère un graphe de jeu sur un processus à pile non étiqueté. Il n'est pas difficile d'adapter la définition pour un graphe de processus à pile étiqueté.

2.3 Complexité borélienne

Considérons un alphabet pouvant être infini, que l'on note A. On munit alors l'ensemble A^ω des mots infinis sur l'alphabet A de la topologie de Cantor, où les ouverts sont les ensembles de la forme $W \cdot A^\omega$ pour un langage $W \subseteq A^*$ de mots finis sur l'alphabet A. La *hiérarchie borélienne finie*, $(\Sigma_1,\Pi_1),(\Sigma_2,\Pi_2),\cdots$, est la tribu engendrée par les ouverts. Plus précisément, elle est donnée par :

- $\Sigma_1 = \{W \cdot A^\omega \mid W \subseteq A^*\}$ est la classe des ensembles ouverts.
- pour tout $n \geq 1$, $\Pi_n = \{\overline{S} \mid S \in \Sigma_n\}$ est la classe des complémentaires des ensembles de Σ_n. En particulier, Π_1 est la classe des ensembles fermés.
- pour tout $n \geq 1$, $\Sigma_{n+1} = \{\bigcup_{i\in\mathbb{N}} S_i \mid \forall i \in \mathbb{N},\ S_i \in \Pi_n\}$ est la classe des unions dénombrables d'ensembles de Π_n.
- symétriquement, pour tout $n \geq 1$, $\Pi_{n+1} = \{\bigcap_{i\in\mathbb{N}} S_i \mid \forall i \in \mathbb{N},\ S_i \in \Sigma_n\}$ est la classe des intersections dénombrables d'ensembles de Σ_n.

Enfin, on note $\mathcal{B}(\Sigma_n)$ la classe des combinaisons booléennes d'ensembles de Σ_n. La figure 2.5 montre les inclusions (strictes) entre ces ensembles.

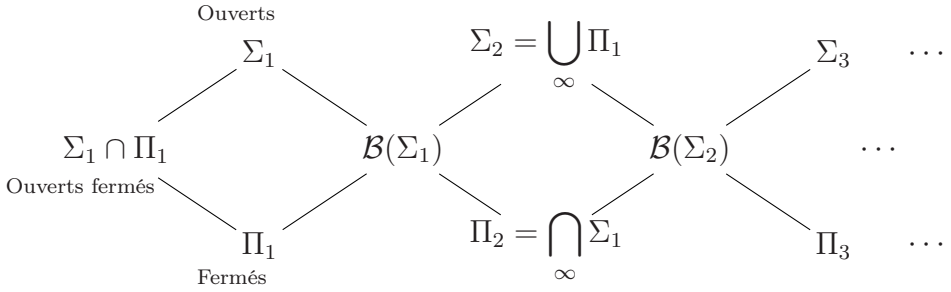


FIG. 2.5 – *La hiérarchie borélienne finie*

2.3.1 Complexité borélienne d'une condition de gain

Considérons un jeu $\mathbb{G} = (\mathcal{G},\Omega)$ joué sur un graphe de jeu $\mathcal{G}$ et équipé d'une condition de gain Ω. La complexité borélienne de la condition de gain Ω est sa complexité borélienne en tant qu'ensemble sur l'alphabet des sommets de $\mathcal{G}$, si Ω est une condition de gain interne, et sur l'alphabet A étiquetant les arcs de $\mathcal{G}$, si Ω est une condition externe.

Précisons maintenant la complexité borélienne de quelques conditions de gain internes pour des jeux joués sur un graphe de sommets V :

1. Considérons une condition d'accessibilité avec un ensemble $F \subseteq V$ de sommets finaux. Cette condition est une condition Σ_1 puisqu'elle est égale à l'ensemble ouvert $(V^*F)V^\omega$.
2. Considérons une condition de Büchi avec un ensemble $F \subseteq V$ de sommets finaux. Cette condition est une condition Π_2 puisqu'elle est égale à l'ensemble $\bigcap_{n\geq 0}[(V^nV^*F)V^\omega]$.
3. Considérons une condition de Muller. Cette condition est une condition $\mathcal{B}(\Sigma_2)$, puisqu'elle s'exprime aisément comme une combinaison booléenne de conditions de Büchi. Cela implique en particulier que les conditions de parité sont des conditions $\mathcal{B}(\Sigma_2)$.
4. Considérons une condition d'explosion pour un jeu sur un graphe de processus à pile. Une telle condition est Π_2. En effet, si pour tout $n \geq 0$, V_n désigne l'ensembles des configurations de hauteur de pile égale à n, la condition de gain pour Eve est égale à $\bigcap_{n\geq 1}[(V^*V_n)V^\omega]$. Elle est donc bien Π_2 en tant qu'intersection dénombrable d'ouverts.
5. Considérons enfin une condition d'explosion stricte pour un jeu sur un graphe de processus à pile. La condition correspondante pour Adam est la condition de répétition étudiée dans [21] : Adam remporte la partie si et seulement si une configuration (de façon équivalente, une hauteur de pile) est visitée infiniment souvent. Si l'on note à nouveau V_n l'ensemble des configurations de hauteur de pile égale à n, pour $n \geq 0$, la condition de gain pour Adam n'est autre qu'une union dénombrable de conditions de Büchi, et est donnée par :

$$\bigcup_{n\geq 1} \bigcap_{m\geq 0} [(V^mV^*V_n)V^\omega]$$

Dès lors, la condition pour Adam est dans Σ_3 et la condition d'explosion stricte est donc dans Π_3.

Mentionnons maintenant le théorème de Martin concernant la déterminaison des jeux munis d'une condition de gain borélienne.

Théorème 2.1 *[58] Tout jeu équipé d'une condition de gain borélienne est déterminé.*

2.3.2 Jeu de Wadge

On introduit maintenant les notions de réduction et de jeu de Wadge, qui permettent de définir une notion de complétude pour les différents niveaux de la hiérarchie borélienne.

Définition 2.16 (Jeu de Wadge) *[81] Soient A et B deux alphabets (pouvant être infinis) et soient $X \subseteq A^\omega$ et $Y \subseteq B^\omega$ deux langages de mots infinis sur ces alphabets. Le* jeu de Wadge $G(X,Y)$ *est un jeu entre deux joueurs, Alice et Bob. Alice commence par choisir une lettre a_0 dans A. Ensuite, Bob choisit un mot fini (pouvant être vide) $b_0 \in B^*$ sur l'alphabet B. Ensuite Alice choisit une nouvelle lettre $a_1 \in A$ et Bob un nouveau mot $b_1 \in B^*$ et ainsi de suite. Un partie revient donc pour Alice à écrire un mot infini $\alpha = a_0 a_1 \cdots$ sur l'alphabet A et, pour Bob, à écrire un mot (fini ou infini) $\beta = b_0 b_1 \cdots$ sur l'alphabet B. Si le mot β est fini, Alice remporte la partie. Sinon, Bob gagne si et seulement si on a : $\alpha \in X \Leftrightarrow \beta \in Y$.*

Intuitivement, la première condition (Bob perd s'il écrit un mot fini) oblige Bob à jouer (c'est-à-dire à jouer un mot non vide) infiniment souvent. La seconde condition traduit le fait que le langage Y capture la complexité du langage X.

On définit sans problème les notions de stratégies et de stratégies gagnantes pour les jeux de Wadge.

Ces jeux sont en fait très fortement reliés (voir la proposition 2.2) à la notion de réduction de Wadge définie ci-dessous.

Définition 2.17 (Réduction de Wadge) *On dit d'un ensemble $X \subseteq A^\omega$ qu'il se* réduit au sens de Wadge *à un ensemble $Y \subseteq B^\omega$, ce que l'on note $X \leq_W Y$, si et seulement s'il existe une fonction continue (pour la topologie de Cantor) $f : A^\omega \to B^\omega$ telle que $X = f^{-1}(Y)$. Dans le cas où $X \leq_W Y$ et $Y \leq_W X$, les ensemble X et Y sont dits* Wadge équivalents, *ce que l'on note $X \equiv_W Y$.*

Proposition 2.2 ([81]) *Bob possède une stratégie gagnante dans $G(X,Y)$ si et seulement si $X \leq_W Y$.*

Donnons quelques exemples de cette réduction.

Exemple 2.1 Considérons le langage A^ω où A est un alphabet non vide. On a alors $A^\omega \leq_w Y$ pour tout ensemble non vide Y. En effet, Bob a une stratégie gagnante dans tout jeu $W(A^\omega,Y)$ qui consiste a écrire un mot dans Y.

Considérons maintenant l'alphabet $A = \{a,b\}$. Soit $X \subseteq A^\omega$ un ensemble fermé et soit $Y = \{a^\omega\} \subseteq A^\omega$. Alors, $X \leq_w Y$. En effet, une stratégie gagnante pour

Bob consiste à jouer a si le mot déjà écrit par Alice est préfixe d'un mot de X. Sinon, il joue b. Une telle stratégie est gagnante puisqu'un mot infini appartient à un ensemble fermé X si et seulement si tout préfixe de ce mot est préfixe d'un mot de X (voir par exemple [70]).

Ces deux exemples soulignent l'importance capitale de l'alphabet sous-jacent sur lequel on considère le langage. En effet, le singleton $\{a^\omega\}$ n'a pas la même complexité selon qu'il est considéré sur un alphabet à une lettre ou à plusieurs lettres.

La hiérarchie de Wadge est en fait un raffinement de la hiérarchie de Borel (pour plus de détails, voir le chapitre sur la hiérarchie de Wagner dans [70]). Une des conséquences est que l'équivalence de Wadge préserve les niveaux de la hiérarchie borélienne.

Proposition 2.3 *Deux ensembles Wadge équivalents appartiennent au même niveau de la hiérarchie de Borel.*

Il est alors naturel de considérer la notion de complétude induite par l'ordre partiel $\leq_W$:

Définition 2.18 *Un ensemble $Y \in \Sigma_n$ est qualifié* d'ensemble Σ_n-complet *si et seulement si pour tout ensemble $X \in \Sigma_n$, on a $X \leq_W Y$. Symétriquement, on définit la notion d'ensemble Π_n-complets.*

Remarque 2.6 Signalons que la notion de complétude n'est pas pertinente pour les classes $\mathcal{B}(\Sigma_n)$, car, dès que $n \geq 2$, ces dernières ne contiennent pas d'ensembles complets (voir par exemple [47]).

Finissons avec un exemple d'ensemble Π_2-complet.

Exemple 2.2 On considère l'alphabet $A = \{a,b\}$ et l'ensemble $X \subseteq A^\omega$ des mots contenant une infinité d'occurrences de la lettre a. L'ensemble X est alors Π_2-complet. En effet $X = \bigcap_{i\geq 0} A^i A^* a A^\omega$, ce qui prouve que X est dans Π_2. Pour la complétude, considérons un ensemble Y dans Π_2 sur un alphabet B. On peut alors écrire Y sous la forme $Y = \bigcap_{i\geq 0} Y_i$ pour une famille $(Y_i)_{i\geq 0}$ d'ouverts que l'on note $Y_i = Z_i B^\omega$.

Dans le jeu $G(Y,X)$, Bob a une stratégie gagnante qui maintient un compteur i qui est initialisé à 0. Si le mot écrit par Alice est dans $Z_i B^*$, Bob joue a et incrémente son compteur d'une unité. Sinon, il joue un b et ne modifie pas le compteur. Dès lors, le mot joué par Bob est toujours infini et contient une infinité de a si et seulement si le mot joué par Alice appartient à Y_i pour tout $i \geq 0$, c'est-à-dire s'il appartient à Y.

2.4 Quelques résultats classiques

On termine ce chapitre en donnant quelques résultats connus concernant les jeux sur des graphes finis ainsi que sur le caractère sans mémoire des stratégies gagnantes pour certaines conditions de gain.

Concernant la déterminaison des jeux munis d'une condition de Muller, on a le résultat suivant.

Théorème 2.2 *[58, 30, 64] Soit $\mathbb{G} = (\mathcal{G},\Omega)$ un jeu de Muller sur un graphe quelconque. Le jeu $\mathbb{G}$ est déterminé. De plus, lorsque le graphe de jeu est fini, on peut calculer l'ensemble des positions gagnantes. Enfin, quelle que soit la nature du graphe de jeu, si la condition de gain Ω est une condition de parité, alors chaque joueur possède une stratégie positionnelle sur son ensemble de positions gagnantes.*

On pourra consulter [77, 87] pour une présentation de ces résultats. Bien qu'elle ne soit nécessaire dans aucune des preuves des résultats présentés dans cette thèse, l'existence de stratégies positionnelles pour les jeux de parité simplifiera notoirement certaines d'entre elles.

Le théorème 2.2 soulève plusieurs questions :

- quelle est la complexité du problème de décision du gagnant dans un jeu sur un graphe fini muni d'une condition d'accessibilité, de Büchi, de parité ou de Muller? Cette question fondamentale a largement été étudiée et reste encore aujourd'hui très active. Nous présentons quelques résultats ci-dessous.
- il existe toujours des stratégies positionnelles pour les jeux de parité. Pour les condition de Muller ce n'est en revanche plus le cas (voir par exemple [87]). Pour quelles autres conditions y a-t-il toujours des stratégies positionnelles? On consultera [40, 42, 27, 39].
- on peut calculer les ensembles de positions gagnantes dans des jeux de Muller sur des graphes fini. Pour des jeux sur des graphes infinis finiment représentables, pour quelles conditions peut-on calculer les ensembles de positions gagnantes. Cette question, dans le cadre des jeux sur des graphes de processus à pile, a été initiée par I. Walukiewicz dans [83] et fait l'objet de cette thèse. Un historique et un panorama de la question est donné dans le chapitre 3.

Pour finir, mentionnons les réponses connues à la première question. On considère donc un graphe de jeu fini $\mathcal{G} = ((V,E),V_{\mathbf{E}},V_{\mathbf{A}})$. Pour ce qui est des conditions d'accessibilité et de Büchi, on a les résultat suivants :

Théorème 2.3 *Soit $\mathbb{G}$ un jeu d'accessibilité sur un graphe de jeu fini $((V,E),V_{\mathbf{E}},V_{\mathbf{A}})$. On peut calculer l'ensemble des positions gagnantes pour Eve dans $\mathbb{G}$ en temps $\mathcal{O}(n+m)$ où $n = |V|$ et $m = |E|$.*

Théorème 2.4 *Soit $\mathbb{G}$ un jeu de Büchi sur un graphe de jeu fini $((V,E),V_{\mathbf{E}},V_{\mathbf{A}})$. On peut calculer l'ensemble des positions gagnantes pour Eve dans $\mathbb{G}$ en temps $\mathcal{O}(n(n+m))$ où $n = |V|$ et $m = |E|$.*

Concernant les condition de parité, le problème est beaucoup plus ouvert. Plus précisément, on a le résultat suivant, où UP est une sous-classe de NP (voir [69, 45] pour plus de détails).

Théorème 2.5 *[45] Décider le gagnant dans un jeu de parité sur un graphe fini est dans* $\mathrm{UP} \cap co\mathrm{UP}$.

On ne connaît pas à ce jour d'algorithme polynomial permettant de décider le gagnant dans un jeu de parité sur un graphe fini. Les meilleurs algorithmes connus sont exponentiels dans le nombre de couleurs intervenant dans la condition de parité.

Théorème 2.6 *[46, 80] Soit $\mathbb{G}$ un jeu de parité à d couleurs sur un graphe de jeu fini $((V,E),V_{\mathbf{E}},V_{\mathbf{A}})$. On peut calculer l'ensemble des positions gagnantes pour Eve dans $\mathbb{G}$ en temps*

$$\mathcal{O}\left(d \cdot m \cdot \left(\frac{n+d}{d}\right)^{\lceil d/2 \rceil}\right),$$

où $n = |V|$ et $m = |E|$.

Remarque 2.7 Il existe deux algorithmes réalisant la complexité présentée ci-dessus. On connaît pour l'algorithme donné dans [46] un exemple où cette complexité est atteinte. En revanche, on ne connaît pas de borne inférieure pour l'algorithme présenté dans [80], et en particulier, on ne sait pas s'il est polynomial.

Chapitre 3

Jeux sur des graphes de processus à pile : petit historique et contributions de la thèse

Donnons maintenant un petit historique des travaux concernant les jeux sur des graphes de processus à pile, en présentant les contributions et la structure de cette thèse.

Les recherches effectuées sur les automate à pile alternant peuvent être considérées comme les premières études concernant les jeux sur des graphes de processus à pile. A ma connaissance, le premier papier qui fait mention d'automate à pile alternant est celui de A.K. Chandra et L.J. Stockmeyer [25], dans lequel est mentionné le résultat suivant : si L est un langage dans DEXPTIME, alors il est accepté par un automate à pile alternant. La réciproque de ce résultat est donnée par R.E. Ladner, R.J. Lipton et L.J. Stockmeyer, dans un article [52] entièrement dédié aux automates à pile alternants et à leurs extensions. Plus précisément, les auteurs y prouvent, entre autre que les langages reconnus par des automates à pile alternants sont exactement ceux acceptés par des machines de Turing déterministes fonctionnant en temps exponentiel. Cependant, on ne peut en déduire un algorithme pour décider le gagnant dans un jeu de parité sur un graphe de processus à pile. En effet, ce problème exprimé en terme d'automate à pile alternant (ou donc de machine de Turing déterministe fonctionnant en temps exponentiel) revient à décider le problème du vide.

Les processus à pile permettent de modéliser de nombreux systèmes. Il est alors naturel d'étudier le problème du model-checking (vérification de modèle) qui consiste à décider si une formule logique est satisfaite par un système. I. Walukiewicz, a établi dans [83, 84], que le problème du model-checking du μ-calcul pour les graphes de processus à pile est un problème DEXPTIME-complet. Il a tout d'abord montré que le problème précédent est polynomialement équivalent à décider le gagnant dans un jeu de parité sur un graphe de processus à pile. Une réduction du dernier problème au problème de décision du gagnant dans un jeu de parité, avec le même nombre de couleurs, mais sur un graphe fini de taille exponentielle, est ensuite donnée. De cette dernière réduction découle la borne inférieure de complexité. La borne supérieure est quant à elle obtenue en simulant une machine de Turing alternante à l'aide d'un jeu d'accessibilité sur un graphe de processus à compteur. Une construction proche de celle de [83, 84] est donnée dans le chapitre 5.

Une résolution différente d'un sous-problème du précédent est donnée par A. Bouajjani, J. Esparza et O. Maler dans [11]. Les auteurs considèrent le problème de calcul du pre^* d'un ensemble régulier de configuration dans un processus à pile alternant. Ce problème n'est autre que celui du calcul des positions gagnantes dans un jeu d'accessibilité sur un graphe de processus à pile, où l'ensemble des configurations finales est un ensemble régulier (ce jeu généralise donc les jeux d'accessibilité au sens où nous les avons définis). La principale motivation pour étudier ce problème est qu'il permet de résoudre le problème du model-checking, pour des formules du μ-calcul sans alternance et de la logique EF, pour les graphes de processus à pile. A la différence de ce qui était fait dans [83, 84], la construction donnée dans [11] est symbolique, c'est-à-dire qu'elle donne un $\mathcal{P}$-automate alternant qui reconnaît l'ensemble des positions gagnantes pour Eve ou, en terme de logique, l'ensemble des configurations satisfaisant une formule donnée.

Une nouvelle approche a été ensuite développée par O. Kupferman et M. Vardi dans [50]. Les auteurs montrent comment le problème de décision du gagnant dans un jeu de parité sur un graphe de processus à pile peut être réduit à décider le vide pour un automate d'arbre bidirectionnel alternant équipé d'une condition de parité. M. Vardi ayant montré dans [79] que ce dernier problème était Dexptime, les auteurs obtiennent alors facilement les résultats donnés par I. Walukiewicz dans [83, 84].

En généralisant les techniques de [11], T. Cachat a donné dans [16] une construction d'un $\mathcal{P}$-automate alternant reconnaissant l'ensemble des positions gagnantes pour Eve dans un jeu de Büchi sur un graphe de processus à pile. Décider le gagnant depuis une configuration quelconque pouvait être réalisé en modifiant les algorithmes donnés par I. Walukiewicz et par O. Kupferman et M. Vardi, mais le principal intérêt du résultat de T. Cachat est de donner une construction uniforme et de montrer au passage que les ensembles de positions gagnantes pour les deux joueurs sont réguliers.

Une question restait alors en suspend, à savoir donner une construction uniforme pour les conditions de parité. Nous avons avec T. Cachat répondu indépendamment à cette question dans [75] et [18], en montrant comment construire un $\mathcal{P}$-automate alternant reconnaissant l'ensemble des positions gagnantes pour Eve. Dans [18], T. Cachat montre que le résultat est encore vrai lorsque l'on joue sur des graphes préfixes reconnaissables qui généralisent les graphes de processus à pile. Dans [75], j'ai également montré que l'on peut adapter la construction à des conditions plus générales, à savoir les conditions ω-régulières sur les états (qui généralisent les conditions de Muller). Dans le chapitre 4, nous exposons les résultats de [75] en les étendant très largement.

Concernant les jeux d'accessibilité et de Büchi pour des graphes de processus à pile, mentionnons également que plusieurs études ont été réalisées récemment dans le cas où l'on considère non pas un processus à pile, mais une *recursive state machine*. Ce modèle est équivalent à celui des processus à pile, mais il est plus pratique pour modéliser des programmes [23]. Dans [3, 2, 34], R. Alur, P. Madhusudan, S. La Torre et K. Etessami s'intéressent aux jeux d'accessibilité et de Büchi ainsi qu'à la question de savoir, s'il existe dans ces jeux, des stratégies particulières, dites *modulaires*. Signalons enfin que le modèle des *recursive state machine* permet une

analyse plus fine de la complexité, analyse qui peut cependant être également menée dans le cadre des processus à pile.

Pour les sous-classes des processus à pile que forment les BPA et les processus à compteur, on peut également étudier les jeux de parité en cherchant des algorithmes plus simples que dans le cas général des processus à pile. Pour les BPA, on modifie les définitions de la partition des configurations et de la fonction de coloriage (car sinon, les jeux obtenus sont triviaux). On donne un premier modèle dans lequel le problème de décision du gagnant pour une condition de parité se réduit polynomialement au même problème pour un jeu sur un graphe fini. Pour un modèle plus complexe, on montre que le même problème est alors DEXPTIME-complet. Pour ce qui est des jeux de parité sur des graphes à compteur, on montre, en adaptant les techniques de [79, 50], que l'on peut décider le gagnant en PSPACE. On donne également une borne inférieure, en montrant que le problème est DP-dur. Une conséquence de ces résultats est que le problème du model-checking du μ-calcul pour un processus à compteur est dans PSPACE. Ces résultats sont présentés dans le chapitre 7.

Nous avons déjà vu dans le chapitre 2, que, d'un point de vue topologique, les conditions de parité, sont des conditions $\mathcal{B}(\Sigma_2)$. Une question naturelle était de donner un exemple de jeu muni d'une condition de gain de complexité borélienne supérieure au deuxième niveau de la hiérarchie borélienne, et pour lequel on peut décider le gagnant. Par ailleurs, les jeux sur des graphes de processus à pile pouvant modéliser des programmes autorisant des appels récursifs, les conditions sur la hauteur de pile s'imposaient alors naturellement. Dans [21], T. Cachat, J. Duparc et W. Thomas, ont considéré les jeux sur des graphes de processus à pile munis de la condition de répétition : Eve gagne si et seulement si une configuration est infiniment souvent répétée. Ils ont montré que cette condition est Σ_3-complète, et que l'on peut décider le gagnant en DEXPTIME. Dans le chapitre 5, nous donnerons une autre preuve de ce résultat, en considérant la condition duale, à savoir la condition d'explosion stricte. Nous montrerons aussi que la condition d'explosion, qui est plus naturelle que les deux précédentes pour ce qui est de la vérification, est également décidable avec la même complexité.

Dans l'optique de la vérification, les conditions de parité permettent de vérifier des conditions régulières, décrites par exemple par une formule logique (LTL, μ-calcul...), tandis que la condition d'explosion permet de vérifier des conditions de bornage de la pile d'appels récursifs. Il est alors naturel de se demander si l'on peut vérifier une combinaison booléenne de ces propriétés. On peut par exemple chercher un contrôleur permettant à un programme de ne pas exploser sa pile d'appels récursifs tout en satisfaisant une condition décrite par une formule du μ-calcul. En terme de jeux, on considère donc des conditions de gain qui sont des combinaisons booléennes de condition de parité et d'explosion. Ce problème est compliqué puisque l'on ne peut utiliser directement les résultats concernant, d'un côté les condition de parité, et de l'autre les conditions d'explosion. Avec A-J. Bouquet et I. Walukiewicz, nous avons proposé une solution pour les combinaisons booléennes de condition de Büchi et d'explosion [13]. Dans le chapitre 8, on présente une autre construction qui utilise une réduction vers un jeu sur un graphe de processus à compteur. Cette dernière permet de donner une solution lorsque la condition de gain est une combinaisons booléennes d'une condition de parité et d'un condition sur la hauteur de pile.

La complexité n'est par contre pas optimale. On donne également dans le chapitre 8 les constructions de [13] mais pas les preuves. Concernant les combinaisons booléennes de condition de parité et d'explosion, une solution, basée sur des techniques proches de [50], a été donnée par H. Gimbert dans [38].

D'un point de vue plus théorique, une autre extension des résultats de [21] consiste à chercher une famille de conditions de gain de complexité borélienne arbitraire finie et qui reste décidable. J'ai répondu à cette question dans [76, 74]. Les résultats sont présentés dans le chapitre 6, dans lequel on donne une famille de conditions de gain ainsi que la complexité borélienne des conditions qui la composent. On explique ensuite comment décider le gagnant dans les jeux équipés d'une telle condition. On montre enfin que le problème de décision associé est non élémentaire et Elementary-dur.

Enfin, récemment, R. Alur, K. Etessami et P. Madhusudan ont introduit la logique Caret, une extension de ltl, permettant d'exprimer des propriétés naturelles des systèmes avec appels récursifs. Par ailleurs, dans [4], R. Alur et P. Madhusudan ont introduit la notion de Vpa qui permet de décrire les langages des modèles des formules de Caret. Avec C. Löding et P. Madhusudan, nous avons montré dans [56] que les jeux sur des graphes de Vpp équipés d'une condition de gain externe décrite par un Vpa non déterministe de Büchi sont décidables. Pour cela, nous avons proposé une procédure de déterminisation des Vpa. Ce modèle déterministe permet alors de réduire le problème précédent à un jeu muni d'une condition de parité en escalier. Ces résultats sont présentés dans les chapitres 8 et 5.

Chapitre 4

Ensembles de positions gagnantes

Sommaire

Dans [83, 84], I. Walukiewicz propose une solution optimale pour les jeux sur des graphes de processus à pile munis d'une condition de parité. La solution proposée permet en fait de décider si les configurations de la forme $(p,\perp)$ sont gagnantes. Se restreindre à l'étude des configurations de pile vide n'est pas une perte de généralité, puisqu'il n'est pas difficile de réduire le problème de décision du gagnant depuis une position quelconque au même problème depuis une configuration de pile vide (on définit un nouveau jeu, dans lequel le début de partie est consacré à atteindre la configuration de pile non vide à laquelle on s'intéresse et l'on simule ensuite le jeu précédent). Cependant, bien que l'on puisse déterminer le gagnant depuis n'importe quelle configuration, une question demeure : comment représenter, si cela est possible, les ensembles de positions gagnantes? Bien sûr, selon les conditions de gain considérées, cette question appelle des réponses différentes.

Dans le cas des conditions d'accessibilité, A. Bouajjani, J. Esparza et O. Maler ont prouvé que les ensembles de positions gagnantes sont réguliers et que l'on peut construire un $\mathcal{P}$-automate alternant reconnaissant les positions gagnantes [11]. Ce résultat a ensuite été étendu par T. Cachat au cas des conditions de Büchi [16]. Enfin, pour les conditions de parité, le résultat est donné dans [18, 75]. On peut également signaler dès maintenant qu'un exemple d'une condition de gain pour laquelle les ensembles de positions gagnantes ne sont pas réguliers est donné dans le chapitre 6 et dans [76, 74].

Dans ce chapitre, nous donnons la construction proposée dans [75] en l'adaptant à un cadre bien plus général que celui des conditions de parité. Dans le paragraphe 4.1, on introduit deux notions importantes pour la suite : les invariances par translations, qui sont des propriétés des conditions de gain, et les ensembles de retours qui permettent de construire des automates reconnaissant les ensembles de positions gagnantes lorsque ceux-ci sont réguliers. Dans le paragraphe 4.2, on s'intéresse aux conditions de gains invariantes par translations, qui conduisent à des jeux dans lesquels les ensembles de positions gagnantes sont réguliers. Outre la construction d'un $\mathcal{P}$-automate alternant reconnaissant les positions gagnantes, on décrit également les stratégies gagnantes. Enfin, dans le paragraphe 4.3, on généralise les constructions précédentes afin de décrire une classe générale de conditions de gain conduisant à des ensembles réguliers de positions gagnantes.

Dans ce chapitre, nous nous plaçons dans le cadre des jeux sur des graphes de processus à pile non étiqueté. Ainsi, les parties seront vues comme des éléments de V^∞, si V désigne l'ensemble des sommets du graphe de jeu. De plus, les conditions seront alors des conditions de gain internes.

4.1 Quelques définitions et un peu d'intuition

4.1.1 Remarques préliminaires

Dans un premier temps, considérons un jeu sur un graphe de processus à pile muni d'une condition de parité. Appelons Γ l'alphabet de pile et $\perp$ le symbole de fond de pile. Imaginons qu'Eve possède une stratégie gagnante φ depuis une position $(p,a\perp)$. Supposons qu'en plus d'être remportées par Eve, les parties jouées selon φ, ne visitent jamais de configurations de pile vide, c'est-à-dire que le a qui se trouve au fond de la pile n'est jamais dépilé. Pour tout mot $u \in (\Gamma \setminus \{\perp\})^*$, on déduit alors de φ une stratégie gagnante depuis $(p,au\perp)$. En effet, pour gagner depuis $(p,au\perp)$, Eve peut se contenter d'oublier tout ce qui se trouve sous le a et de jouer alors selon φ : une partie ainsi obtenue ne dépile jamais le a et est égale à une partie depuis $(p,a\perp)$, où Eve respecte φ, et dans laquelle tous les contenus de pile ont été augmentés de u en leur fond.

Symétriquement, on peut mener le même raisonnement pour les positions de la forme $(p,a\perp)$, depuis lesquelles Adam possède une stratégie gagnante lui assurant en plus que le a n'est jamais dépilé.

Cependant, il existe en général des positions gagnantes pour Eve (et symétriquement pour Adam) depuis lesquelles cette dernière ne peut, tout en remportant la victoire, faire en sorte que la pile ne soit jamais vide. Pour cela, il suffit d'imaginer

un jeu muni d'une condition de Büchi où tous les états sont finaux et appartiennent à Adam. De plus, si Adam peut empiler et dépiler n'importe quelle lettre, les parties sont quelconques, bien que toutes gagnées par Eve.

4.1.2 Deux propriétés sur les conditions de gain

Imaginons maintenant un jeu sur un graphe de processus à pile avec la condition de gain suivante : Eve remporte la partie à condition qu'à aucun moment la pile ne contienne un b. Imaginons de surcroît qu'Eve peut, depuis une position $(p,a\bot)$, faire infiniment la boucle $(p,a\bot)\cdot(p,aa\bot)\cdot(p,a\bot)$. Ainsi, Eve possède une stratégie gagnante depuis $(p,a\bot)$ qui ne vide jamais la pile. Cependant, il est clair que toute les positions de la forme $(p,au\bot)$ ne sont pas gagnantes (songez par exemple à $(p,ab\bot)$). Le raisonnement précédent ne tient donc plus et cela pour une raison simple : les parties gagnantes ne sont pas invariantes par ajout d'un mot en fond de pile dans toutes les positions de la partie.

On en vient donc à introduire les définitions et les propriétés suivantes (qui sont illustrées par les figures 4.1 et 4.2) :

Définition 4.1 *Soit Γ un alphabet de pile, de symbole de fond de pile $\bot$, et soit Q un ensemble d'états. Soit u un mot sur l'alphabet $(\Gamma\setminus\{\bot\})$. On définit l'application τ_u de $Q\times(\Gamma\setminus\{\bot\})^+\bot$ dans lui-même, par $\tau_u(q,w\bot)=(q,wu\bot)$ pour tout état $q\in Q$ et tout mot $w\in(\Gamma\setminus\{\bot\})^+$. On l'étend alors en un endomorphisme lettre à lettre de $(Q\times(\Gamma\setminus\{\bot\})^+\bot)^*$. Ainsi, si l'on considère un jeu sur un graphe de processus à pile d'alphabet de pile Γ, τ_u associe à toute partie où la pile n'est jamais vidée la partie (valide) obtenue en ajoutant, dans toutes les configurations, u au fond de la pile.*

Définition 4.2 *Soit Γ un alphabet de pile, de symbole de fond de pile $\bot$, et soit Q un ensemble d'états. Soit u un mot sur l'alphabet $(\Gamma\setminus\{\bot\})$. On introduit l'application $\tau_{u^{-1}}$ de $Q\times(\Gamma\setminus\{\bot\})^+u\bot$ dans $Q\times(\Gamma\setminus\{\bot\})^+\bot$, définie par $\tau_{u^{-1}}(q,wu\bot)=(q,w\bot)$ pour tout état $q\in Q$ et tout mot $w\in(\Gamma\setminus\{\bot\})^+$. Ainsi, $\tau_{u^{-1}}$ associe à toute configuration contenant u en fond de pile avec des lettres au-dessus, la configuration obtenue en supprimant u.*

On étend alors $\tau_{u^{-1}}$ en un morphisme de $(Q\times(\Gamma\setminus\{\bot\})^+u\bot)^*$ dans $(Q\times(\Gamma\setminus\{\bot\})^+\bot)^*$. Ainsi, si l'on considère un jeu sur un graphe de processus à pile d'alphabet de pile Γ, $\tau_{u^{-1}}$ associe à toute partie où u est toujours présent dans la pile avec des lettres au-dessus de lui la partie (valide), obtenue en supprimant u du fond de la pile de toutes les configurations.

Définition 4.3 *Soit Ω une condition de gain pour un jeu sur un graphe de processus à pile et soit Γ l'alphabet de pile sous-jacent à Ω. On dit que Ω est* invariante par translations verticales *si et seulement si pour toute partie $\Lambda=v_0v_1v_2\cdots$ de Ω on a :*

1. **Invariance par translation vers le haut.** *Si Λ ne visite pas de configuration de pile vide, pour tout mot fini $u\in\Gamma^*$ ne contenant pas le fond de pile, $\tau_u(\Lambda)$ est dans Ω.*

2. **Invariance par translation vers le bas.** *Pour tout mot fini $u \in \Gamma^*$ ne contenant pas le fond de pile, et tel que pour toute position $v_i = (p_i,u_i\perp)$ visitée dans Λ, u est un suffixe strict de u_i, $\tau_{u^{-1}}(\Lambda)$ est dans Ω.*

Les figures 4.1 et 4.2 donnent une interprétation graphique de ces invariances.

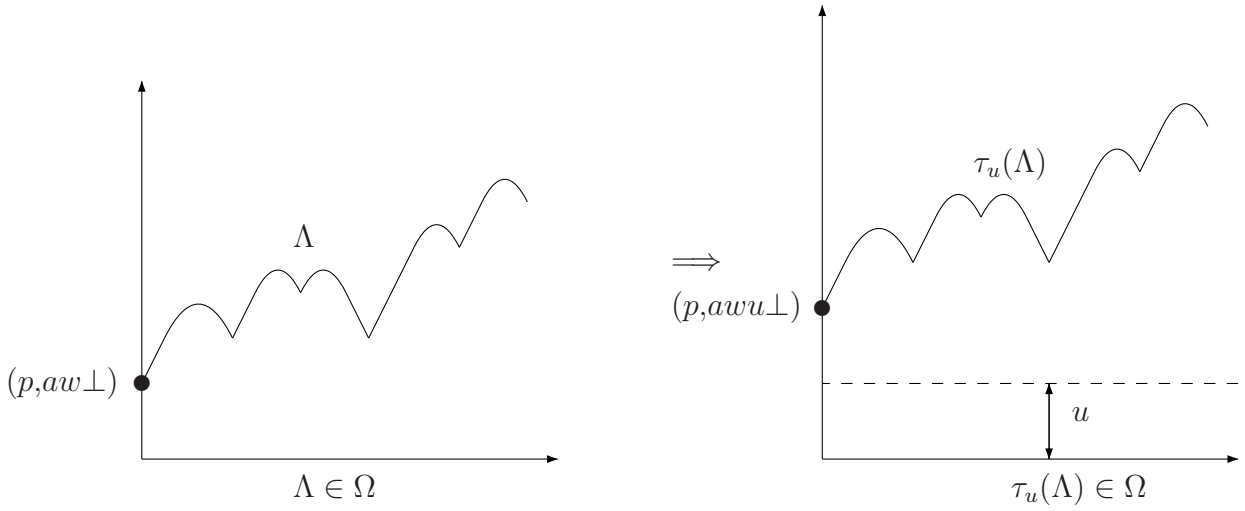


FIG. 4.1 – *Invariance par translation vers le haut*

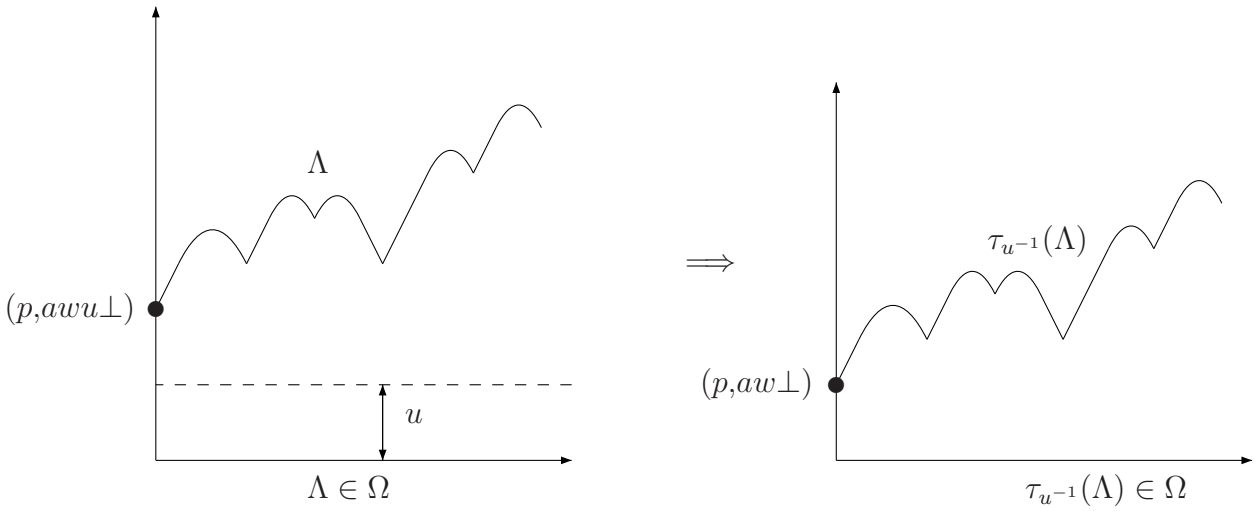


FIG. 4.2 – *Invariance par translation vers le bas*

Par la suite, nous considérerons des conditions de gain qui sont des combinaisons booléennes d'autres conditions de gain. Une question naturelle est alors celle de la clôture par les opérations booléennes de l'ensemble des conditions invariantes par translations verticales. On a le résultat suivant.

Proposition 4.1 *L'ensemble des conditions de gain invariantes par translations verticales est une algèbre de Boole.*

Preuve. Les fermetures par union et intersection sont immédiates. Considérons la fermeture par complémentation et remarquons au préalable le fait suivant : si Λ' est un translaté de Λ vers le haut (respectivement vers le bas), alors Λ est un translaté de Λ' vers le bas (respectivement vers le haut). Soit Ω une condition de gain invariante par translations verticales et soit $\overline{\Omega}$ la condition complémentaire de Ω. Supposons qu'il existe une partie Λ dans $\overline{\Omega}$ et un translaté Λ' de Λ tel que $\Lambda' \notin \overline{\Omega}$. On a alors $\Lambda' \in \Omega$, et comme Λ est un translaté de Λ', $\Lambda \in \Omega$ par invariance de Ω par translations verticales. D'où une contradiction et la fermeture par complémentation. □

Exemple 4.1 Il est facile de voir que toute condition de gain qui ne considère pas le contenu de la pile, c'est-à-dire qui ne dépend que de la suite des états de contrôle, est invariante par translations horizontales. C'est le cas en particulier des conditions d'accessibilité, de Büchi, de parité, ou de Muller portant sur les états de contrôle, mais aussi de conditions plus exotiques qui peuvent, par exemple demander, que la suite des états de contrôle appartienne à un langage (quelconque).

Dès que les conditions d'accessibilité, de Büchi, de parité ou de Muller portent sur le contenu de la pile et non plus sur les états de contrôle, l'invariance par translations verticales n'est plus vérifiée. C'est le cas par exemple d'une condition d'accessibilité demandant d'atteindre une configuration de hauteur de pile divisible par 7.

D'autres conditions naturelles pour les jeux sur un graphe de processus à pile sont celles concernant le bornage (ou le non-bornage) de la pile. Elles vérifient l'invariance par translations verticales. En revanche, les conditions plus générales qui étudient la limite de la pile ne sont pas invariantes par translations verticales.

Dans la suite, on va s'intéresser à la composition des stratégies. Sans entrer dans le détail, il s'agit de suivre une stratégie, jusqu'à atteindre une certaine position à partir de laquelle on suit une autre stratégie qui assure la victoire. L'idée sous-jacente est la suivante : si l'on est capable de forcer à atteindre une position gagnante, alors la position de départ est gagnante. C'est le principe de base de la notion de point fixe (ou d'attracteur) utilisée pour résoudre les jeux munis d'une condition d'accessibilité ou de Büchi. La propriété des conditions de gain pour pouvoir enchaîner les parties partielles sans changer l'issue de la partie est leur invariance par ajout ou suppression d'un préfixe de partie. Plus formellement, on définit la propriété suivante.

Définition 4.4 *Soit Ω une condition de gain pour un jeu sur un graphe de processus à pile, de sommets V. On dit que Ω est* invariante par translations horizontales *si et seulement si, pour toute partie Λ dans Ω, on a :*

1. **Invariance par translation vers la gauche.** *Tout suffixe de Λ est dans Ω.*
2. **Invariance par translation vers la droite.** *Toute partie de $V^*\Lambda$ est dans Ω.*

Les figures 4.3 et 4.4 donnent une interprétation graphique de ces invariances.

Pour la clôture booléenne des conditions invariantes par translations horizontales, on établit, de la même façon que la proposition 4.1, le résultat suivant.

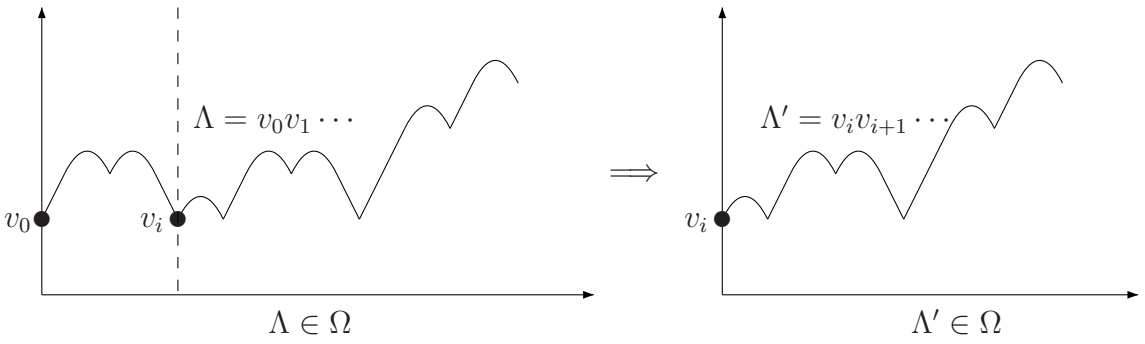


FIG. 4.3 – *Invariance par translation vers la gauche*

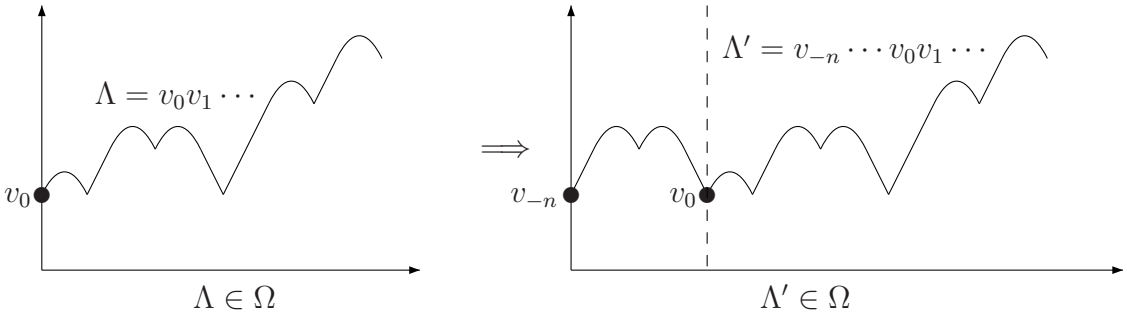


FIG. 4.4 – *Invariance par translation vers la droite*

Proposition 4.2 *L'ensemble des conditions de gain invariantes par translations horizontales est une algèbre de Boole.*

Exemple 4.2 Toute condition de gain ne parlant que d'un comportement à l'infini est invariante par translations horizontales. C'est la cas par exemple des conditions sur l'ensemble des états infiniment répétés comme les conditions de Büchi, de parité, ou plus généralement de Muller. Demander qu'un contenu de pile soit infiniment souvent répété vérifie également l'invariance par translations horizontales.

Les conditions concernant le bornage (ou le non-bornage) de la pile ainsi que celles portant sur la limite de la pile vérifient l'invariance par translations horizontales.

En revanche, la condition d'accessibilité (respectivement la condition de sûreté) n'est pas invariante par translation vers la gauche (respectivement vers la droite).

4.1.3 Jeux conditionnés et ensembles de retours

Comme nous l'avons vu dans les remarques préliminaires, il est naturel de s'intéresser à la question suivante : Eve peut-elle gagner tout en assurant que la pile ne soit jamais vidée? Plus exactement, la question est de savoir si, pour une lettre a en sommet de pile, Eve peut gagner tout en assurant que ce a ne sera jamais dépilé. Si cela n'est pas possible, Eve aura tout intérêt à jouer de la façon suivante : soit forcer le gain sans dépiler a, soit, si elle est contrainte à dépiler a, atteindre en contrepartie un état favorable. Pour formaliser ces intuitions, introduisons les notions de *jeu conditionné* et d'*ensemble de retours*.

Définition 4.5 (Jeu conditionné) *Soit $\mathcal{G} = (G,V_{\mathbf{E}},V_{\mathbf{A}})$ un graphe de jeu sur un graphe $G = (V,E)$ engendré par un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$. Soit $\mathbb{G} = (\mathcal{G},\Omega)$ un jeu sur $\mathcal{G}$. Pour tout sous-ensemble R de Q, on considère une nouvelle condition de gain, notée $\Omega(R)$ et définie par :*

$$\Omega(R) = (\Omega \setminus V^*(Q \times \{\bot\})V^\infty) \cup (V \setminus (Q \times \{\bot\}))^*(R \times \{\bot\})V^\infty$$

En d'autres termes, les parties gagnantes pour $\Omega(R)$ sont :

- *les parties de Ω dans lesquelles aucune configuration de pile vide n'est visitée.*
- *les parties qui visitent une configuration de pile vide et telles que l'état de contrôle, dans la première configuration de pile vide visitée, appartient à R.*

Le jeu $\mathbb{G}(R) = (\mathcal{G},\Omega(R))$ est appelé jeu conditionné *par R.*

Définition 4.6 (Ensemble de retours) *Soit un jeu $\mathbb{G} = (\mathcal{G},\Omega)$ sur un graphe de jeu engendré par un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$. Soit $p \in Q$ un état de contrôle et soit $a \in \Gamma$ une lettre différente de $\bot$. Soit $R \subseteq Q$, un sous-ensemble d'états de contrôle. On dit que R est un* ensemble de retours *pour (p,a) dans $\mathbb{G}$, si Eve possède une stratégie gagnante dans le jeu $\mathbb{G}(R)$ depuis $(p,a\bot)$. On note $\mathcal{R}(p,a)$ l'ensemble des ensembles de retours pour (p,a).*

On a alors facilement la propriété suivante.

Propriété 4.1 *Soit un jeu $\mathbb{G} = (\mathcal{G},\Omega)$ sur un graphe de jeu engendré par un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$. Soit $p \in Q$ et soit $a \in \Gamma$. Si R est un ensemble de retours pour (p,a), et si $R' \supseteq R$, alors R' est un ensemble de retours pour (p,a).*

Preuve. En effet, si $R \subseteq R'$, $\Omega(R) \subseteq \Omega(R')$. □

Remarque 4.1 La propriété 4.1 implique que les ensembles d'ensembles de retours $\mathcal{R}(p,a)$ sont uniquement déterminés par leurs éléments minimaux pour l'inclusion.

Il s'avère que la totalité des résultats où interviendront des ensembles de retours seront encore valides lorsque l'on se restreint aux ensembles de retours minimaux pour l'inclusion. L'intérêt d'une telle restriction, outre le fait qu'elle permet des représentations plus compactes, est qu'elle conduit à des algorithmes toujours plus rapides. Dans le cas où l'on ne se restreint pas aux ensembles minimaux, il peut y avoir un nombre exponentiel d'ensembles de retours (par exemple, si $\emptyset \in \mathcal{R}(p,a)$, tout sous-ensemble d'états est un ensemble de retours). Cependant, il est important de noter qu'il peut exister un nombre exponentiel d'ensembles de retours minimaux.

Pour illustrer cela, on va construire un jeu sur un graphe de processus à pile, dont les parties se déroulent en deux temps. Tout d'abord, Eve choisit un sous-ensemble de n indices parmi $2n$ possibles (sous-ensemble qui sera ensuite associé à un sous-ensemble d'états), qu'elle code en empilant des lettres. Une fois ce choix effectué, Adam dépile jusqu'à choisir un des indices donnés par Eve, qui détermine alors l'état qui sera atteint une fois la pile vidée. On ajoute des états et une condition d'accessibilité pour contraindre le comportement des joueurs (entre autre pour obliger Eve à choisir des éléments distincts et Adam à choisir un indice). Plus formellement, on considère un entier $n \geq 1$ et le processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ suivant :

- $Q = \{p_0,\ldots,p_n\} \cup \{q_1,\ldots,q_{2n}\} \cup \{w,l\}$.
- $\Gamma = \{a_1,\cdots,a_{2n},\#,\bot\}$.
- Δ est définie par :
  - $push(p_{j+1},a') \in \Delta(p_j,a)$ pour $0 \leq j \leq n-1$, $a \in \Gamma$ et $a' \in \{a_1,\cdots,a_{2n}\}$. Dans cette phase, Eve choisit le sous-ensemble d'indices.
  - $\Delta(p_n,a_i) = \{pop(p_n),pop(q_i)\}$ pour $1 \leq i \leq 2n$. Adam peut choisir l'indice i (en choisissant q_i comme nouvel état de contrôle) ou non.
  - $\Delta(p_n,\#) = \{skip(w)\}$ et $\Delta(w,\#) = \{skip(w)\}$. Ces deux règles forcent Adam a choisir un indice.
  - $\Delta(q_i,a_i) = \{skip(l)\}$ pour $1 \leq i \leq 2n$ et $\Delta(l,a) = \{skip(l)\}$ pour tout $a \in \Gamma$. Ces règles forcent Eve à choisir des indices tous distincts.
  - $\Delta(q_i,a_j) = \{pop(q_i)\}$ pour tout $i \neq j$. Une fois l'indice choisi, et donc l'état fixé, Adam vide la pile sans changer d'état de contrôle.

On considère le jeu $\mathbb{G}$ sur le graphe engendré par $\mathcal{P}$ muni de la partition $Q_{\mathbf{E}} = \{p_0,\ldots,p_n\}$ et $Q_{\mathbf{A}} = Q \setminus Q_{\mathbf{E}}$, et équipé d'une condition d'accessibilité, où le seul état final est w.

On vérifie alors facilement que $\mathbb{G}$ correspond bien à la description informelle précédemment donnée. Les ensembles de retours minimaux pour $(p_0,\#)$ sont exactement les sous-ensembles de $\{q_1,\ldots,q_{2n}\}$ à n éléments. Ils sont donc en nombre exponentiel, puisqu'il y en a $\frac{(2n)!}{n!^2}$

Exemple 4.3 Afin d'illustrer la notion d'ensemble de retours, introduisons un jeu de parité sur un graphe de processus à pile, qui sera également étudié dans les paragraphes suivants. On considère donc un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$, une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ des états de contrôle, et une fonction de coloriage ρ donnés par :

- $Q = \{p,q,r\}$: $Q_{\mathbf{E}} = \{p,q\}$ et $Q_A = \{r\}$.
- $\Gamma = \{a,b,\bot\}$.
- la relation Δ est donnée par :
  - $\Delta(p,a) = \{pop(q),push(r,a)\}$.
  - $\Delta(q,a) = \{pop(p),push(r,a)\}$.
  - $\Delta(r,a) = \{pop(p),pop(r)\}$.
  - $\Delta(p,b) = \{pop(p),push(p,a)\}$.
  - $\Delta(q,b) = \{pop(p),push(r,a)\}$.
  - $\Delta(r,b) = \{pop(p),push(q,a)\}$.

  - $\Delta(p,\bot) = \{push(p,a),push(r,b)\}$.
  - $\Delta(q,\bot) = \{push(q,a),push(r,a)\}$.
  - $\Delta(r,\bot) = \{push(q,b),push(r,a)\}$.
- $\rho(p) = 0$, $\rho(q) = 2$ et $\rho(r) = 1$.

On souhaite déterminer les ensembles $\mathcal{R}(p,a)$, $\mathcal{R}(p,b)$, $\mathcal{R}(q,a)$, $\mathcal{R}(q,b)$, $\mathcal{R}(r,a)$ et $\mathcal{R}(r,b)$. Pour cela, on va raisonner *à la main* puisque nous n'avons (pas encore) proposé de méthode générique pour calculer ces ensembles. Du fait de la remarque 4.1, on va se contenter de déterminer les ensembles de retours minimaux pour décrire les ensembles $\mathcal{R}(\dots)$.

Appelons $\mathcal{G}$ le graphe de jeu engendré par $\mathcal{P}$ et la partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$.

- $\mathcal{R}(p,a)$: La figure 4.5 donne la structure de $\mathcal{G}$ autour de $(p,a\bot)$. Dans cette figure (et dans celles qui suivent), les arcs partant de configurations de pile vide ne sont pas représentés puisque dans un jeu conditionné, les parties arrivant dans de tels sommets se terminent.

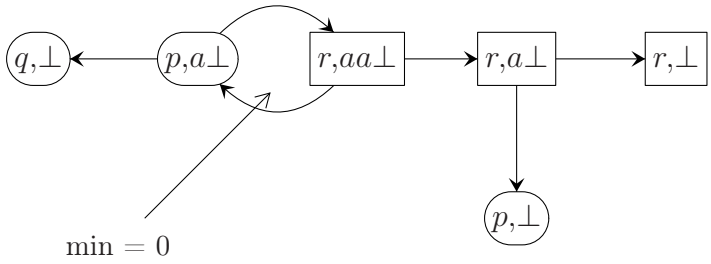


FIG. 4.5 – *Structure de $\mathcal{G}$ autour de $(p,a\bot)$*

  On en déduit que $\mathcal{R}(p,a) = \{\{q\},\{p,r\}\}$ (si l'on ne garde que les éléments minimaux). En effet, Eve gagne dans $\mathbb{G}(\{q\})$. Par ailleurs, elle ne peut éviter que la pile ne soit vidée, donc $\emptyset \notin \mathcal{R}(p,a)$. Enfin, en jouant depuis $(p,a\bot)$ toujours vers $(r,aa\bot)$, elle gagne dans $\mathbb{G}(\{p,r\})$, sans toutefois avoir de stratégie gagnante ni dans $\mathbb{G}(\{p\})$ ni dans $\mathbb{G}(\{r\})$. D'où le résultat.
- $\mathcal{R}(p,b)$: La figure 4.6 décrit la structure de $\mathcal{G}$ autour de $(p,b\bot)$.
  On en déduit que $\mathcal{R}(p,b) = \{\{p\}\}$ (si l'on ne garde que les éléments minimaux). En effet, on remarque tout d'abord qu'Adam peut toujours forcer le dépilement du a vers la configuration $(p,\bot)$. Dès lors, tout ensemble de retours doit contenir p. Par ailleurs, Eve peut forcer le dépilement du a vers la configuration $(p,\bot)$. Le singleton $\{p\}$ est donc un ensemble de retours possible pour (p,a). D'où le résultat.
- $\mathcal{R}(q,a)$: La figure 4.7 décrit la structure de $\mathcal{G}$ autour de $(q,a\bot)$.
  On en déduit que $\mathcal{R}(q,a) = \{\{p\}\}$ (si l'on ne garde que les éléments minimaux). En effet, Eve peut atteindre la configuration $(p,\bot)$ ou la configuration $(r,aa\bot)$ depuis $(q,a\bot)$. Mais depuis $(r,aa\bot)$, Adam peut forcer à atteindre $(p,\bot)$ (et d'autres configurations de pile vide). Eve ne peut donc assurer autre chose de mieux qu'atteindre $(p,\bot)$.

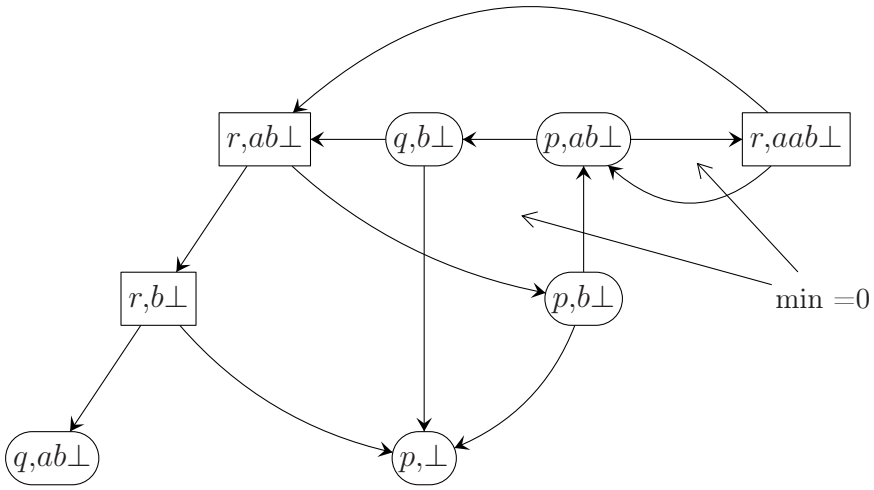


FIG. 4.6 – *Structure de $\mathcal{G}$ autour de $(p,b\bot)$*

– $\mathcal{R}(q,b)$: La figure 4.8 décrit la structure de $\mathcal{G}$ autour de $(q,b\bot)$.
On en déduit que $\mathcal{R}(q,b) = \{\{p\}\}$ (si l'on ne garde que les éléments minimaux). En effet, Eve peut atteindre la configuration $(p,\bot)$ ou la configuration $(r,ab\bot)$ depuis $(q,b\bot)$. Mais depuis $(r,ab\bot)$, Adam peut forcer à atteindre $(p,\bot)$ (et d'autres configurations de pile vide). Eve ne peut donc assurer autre chose de mieux qu'atteindre $(p,\bot)$.

– $\mathcal{R}(r,a)$: La figure 4.9 décrit la structure de $\mathcal{G}$ autour de $(r,a\bot)$.
On en déduit que $\mathcal{R}(r,a) = \{\{p,r\}\}$ (si l'on ne garde que les éléments minimaux).

– $\mathcal{R}(r,b)$: La figure 4.10 décrit la structure de $\mathcal{G}$ autour de $(r,b\bot)$.
On en déduit que $\mathcal{R}(r,b) = \{\{p\}\}$ (si l'on ne garde que les éléments minimaux). En effet, Adam peut forcer le dépilement vers $(p,\bot)$. De plus, si depuis $(r,b\bot)$ il décide de ne pas dépiler, il atteint le sommet $(q,ab\bot)$ depuis lequel Eve peut atteindre $(p,b\bot)$ et ensuite gagner ou dépiler vers $(p,\bot)$.

4.2 Cas particulier des conditions invariantes par translations

Dans tout ce paragraphe, on se donne un jeu $\mathbb{G} = (\mathcal{G},\Omega)$ sur un graphe de jeu engendré par un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ et une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de Q. On suppose de plus que Ω est invariante par translations verticales et horizontales, et que Ω est borélienne. Cela implique qu'en particulier $\Omega(R)$ est borélienne pour tout ensemble R, et donc que les jeux conditionnés intervenant dans la définition des ensembles de retours sont déterminés. Les ensembles de retours sont alors correctement définis.

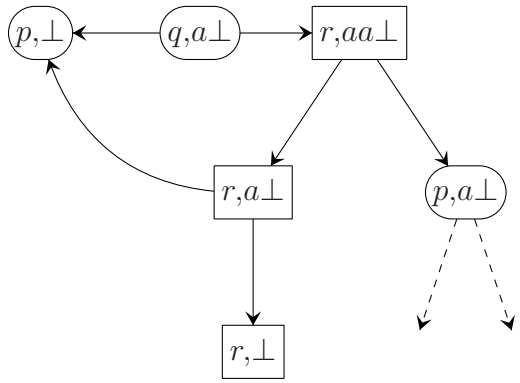


FIG. 4.7 – *Structure de $\mathcal{G}$ autour de $(q,a\bot)$*

4.2.1 Ensembles de retours et positions gagnantes

On commence par le cas particulier déjà évoqué, où Eve peut gagner sans dépiler.

Lemme 4.1 *Soit p un état de contrôle de Q et soit a une lettre de Γ différente de $\bot$. Si $\emptyset \in \mathcal{R}(p,a)$, alors pour tout mot u sur l'alphabet $(\Gamma \setminus \{\bot\})$, $(p,au\bot)$ est une position gagnante pour Eve dans le jeu $\mathbb{G} = (\mathcal{G},\Omega)$.*

Preuve. Soit $\varphi_\emptyset$ une stratégie gagnante pour Eve dans le jeu $\mathbb{G}(\emptyset)$ depuis $(p,a\bot)$. On définit à partir de $\varphi_\emptyset$ une stratégie φ pour Eve dans $\mathbb{G}$ depuis $(p,au\bot)$. Soit $\Lambda = v_0v_1v_2\cdots v_n$ une partie partielle. On définit alors $\varphi(\Lambda)$ de la façon suivante :

- si toute configuration v_i apparaissant dans Λ est de la forme $v_i = (p_i,u_iu\bot)$, où u_i est non vide, on définit alors $\varphi(\Lambda) = \tau_u(\varphi_\emptyset(\tau_{u^{-1}}(\Lambda)))$.
- sinon, $\varphi(\Lambda)$ n'est pas défini.

On montre que si Λ est une partie partielle dans $\mathbb{G}$, commençant en $(p,au\bot)$, où Eve respecte φ, alors toutes les configurations visitées dans Λ sont de la forme $v_i = (p_i,u_iu\bot)$, où u_i est non vide (en particulier, cela implique que si Eve respecte φ, $\varphi(\Lambda)$ est toujours défini) et de plus, $\tau_{u^{-1}}(\Lambda)$ est une partie partielle dans $\mathbb{G}(\emptyset)$ commençant en $(p,a\bot)$ où Eve respecte $\varphi_\emptyset$. Pour cela, on procède par récurrence :

- la propriété est vraie si $\Lambda = (p,au\bot)$.
- supposons maintenant qu'elle soit vraie pour une partie partielle Λ, et montrons qu'elle l'est encore pour un partie Λ' prolongeant Λ d'une configuration. Commençons par le cas où Λ se termine dans une position d'Eve. Comme, par hypothèse de récurrence, $\tau_{u^{-1}}(\Lambda)$ est une partie dans $\mathbb{G}(\emptyset)$, où Eve respecte $\varphi_\emptyset$, $\varphi_\emptyset(\tau_{u^{-1}}(\Lambda))$ est une configuration où la pile n'est pas vide et dès lors

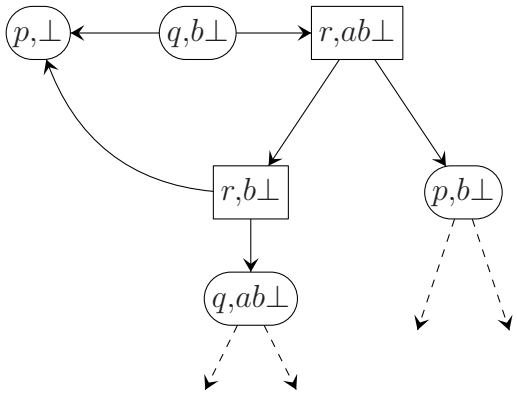


FIG. 4.8 – *Structure de $\mathcal{G}$ autour de* $(q,b\bot)$

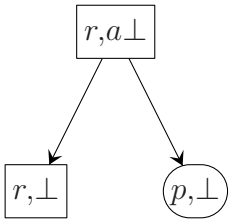


FIG. 4.9 – *Structure de $\mathcal{G}$ autour de* $(r,a\bot)$

$\varphi(\Lambda) = \tau_u(\varphi_\emptyset(\tau_{u^{-1}}(\Lambda)))$ est de la forme $(p_i,u_iu\bot)$, où u_i est non vide. De plus, il est clair que $\tau_{u^{-1}}(\Lambda')$ est une partie partielle dans $\mathbb{G}(\emptyset)$, où Eve respecte $\varphi_\emptyset$. Considérons le cas où Λ se termine par une position d'Adam. Les dernières positions de Λ et de $\tau_{u^{-1}}(\Lambda)$ ont le même état de contrôle et le même sommet de pile. Dès lors, la transition appliquée par Adam une fois Λ jouée est également applicable dans la dernière position de $\tau_{u^{-1}}(\Lambda)$. Comme, par hypothèse de récurrence, $\tau_{u^{-1}}(\Lambda)$ est une partie dans $\mathbb{G}(\emptyset)$ où Eve respecte $\varphi_\emptyset$, le prolongement de $\tau_{u^{-1}}(\Lambda)$ ainsi obtenu est aussi une partie partielle conforme à $\varphi_\emptyset$. La configuration ainsi atteinte n'est donc pas de pile vide. Dès lors, la dernière configuration de Λ' est bien de la forme $(p_i,u_iu\bot)$, où u_i est non vide.

En conclusion, si Λ est une partie dans $\mathbb{G}$ où Eve respecte φ, alors $\tau_{u^{-1}}(\Lambda)$ est une partie dans $\mathbb{G}(\emptyset)$ depuis $(p,a\bot)$, où Eve respecte $\varphi_\emptyset$. Or, $\varphi_\emptyset$ est gagnante, et donc $\tau_{u^{-1}}(\Lambda)$ est gagnante. Dès lors, par invariance vers le haut, $\Lambda = \tau_u(\tau_{u^{-1}}(\Lambda))$ est également gagnante.

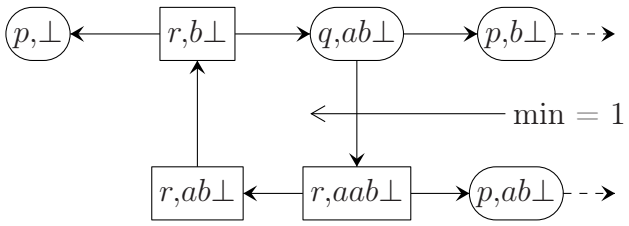


FIG. 4.10 – *Structure de $\mathcal{G}$ autour de* $(r,b\bot)$

Ainsi, φ est une stratégie gagnante pour Eve dans $\mathbb{G}$ depuis la position $(p,au\bot)$. □

De façon symétrique, on a le résultat suivant.

Lemme 4.2 *Soit p un état de contrôle de Q et soit a une lettre de Γ différente de $\bot$. Si $\mathcal{R}(p,a) = \emptyset$, alors pour tout mot u sur l'alphabet $(\Gamma \setminus \{\bot\})$, $(p,au\bot)$ est une position gagnante pour Adam dans le jeu $\mathbb{G} = (\mathcal{G},\Omega)$.*

Que se passe-t-il maintenant pour les couples $(p,a) \in Q \times (\Gamma \setminus \{\bot\})$ pour lesquels $\mathcal{R}(p,a)$ ne contient pas $\emptyset$ ou n'est pas vide? Considérons un tel couple (p,a) et un élément $R \in \mathcal{R}(p,a)$. Eve possède une stratégie gagnante φ_R dans le jeu $\mathbb{G}(R)$ depuis $(p,a\bot)$. A partir de cette stratégie, on peut, comme dans la preuve du lemme 4.1, construire une stratégie (partiellement définie) pour Eve dans $\mathbb{G}$, depuis n'importe quelle configuration de la forme $(p,au\bot)$. Une telle stratégie ne permet pas d'assurer que le a ne sera jamais dépilé et n'est plus définie lorsqu'une telle chose se produit. Cependant, dans le cas où l'on ne dépile pas le a on gagne et sinon on atteint une configuration de la forme $(r,u\bot)$ avec $r \in R$. Pour l'emporter, il faut alors être capable de gagner depuis cette position. Plus formellement on a l'équivalence suivante qui généralise les lemmes 4.1 et 4.2 :

Lemme 4.3 *Soit p un état de contrôle de Q et soit a une lettre de Γ différente de $\bot$. Soit u un mot sur l'alphabet $(\Gamma \setminus \{\bot\})$, alors $(p,au\bot)$ est une position gagnante pour Eve dans le jeu $\mathbb{G}$, si et seulement s'il existe $R \in \mathcal{R}(p,a)$ tel que $(r,u\bot)$ soit gagnant pour Eve dans $\mathbb{G}$ pour tout $r \in R$.*

Preuve. **Implication réciproque :** On commence par démontrer la réciproque, qui est une généralisation des lemmes 4.1 et 4.2. On suppose donc qu'il existe $R \in \mathcal{R}(p,a)$ tel que $(r,u\bot)$ est gagnant dans $\mathbb{G}$ pour tout $r \in R$. Soit φ_R une stratégie gagnante pour Eve depuis $(p,a\bot)$ dans le jeu $\mathbb{G}(R)$. Soient $(\varphi_r)_{r\in R}$ des stratégies gagnantes dans $\mathbb{G}$ depuis les positions $((r,u\bot))_{r\in R}$.

On définit une stratégie φ pour Eve dans le jeu $\mathbb{G}$ depuis la position $(p,au\bot)$. Soit $\Lambda = v_0v_1v_2\cdots v_n$ une partie partielle, $\varphi(\Lambda)$ est alors défini de la façon suivante :

– si toute configuration v_i apparaissant dans Λ est de la forme $(p_i,u_iu\bot)$, où u_i est non vide, on pose $\varphi(\Lambda) = \tau_u(\varphi_R(\tau_{u^{-1}}(\Lambda)))$. Ce cas correspond à une partie Λ où a n'a pas été dépilé.

– s'il existe un entier j tel que $v_j = (r,u\bot)$ pour un certain $r \in R$ et tel que, pour tout $i < j$, v_i est de la forme $(p_i,u_iu\bot)$, on pose $\varphi(\Lambda) = \varphi_r(v_jv_{j+1}\cdots v_n)$. Dans ce cas, on a dépilé le a juste après v_{j-1}.
– sinon $\varphi(\Lambda)$ n'est pas défini.

En raisonnant comme dans la preuve du lemme 4.1, on prouve que si Λ est une partie partielle dans $\mathbb{G}$ commençant en $(p,au\bot)$, où Eve respecte φ, et telle que toutes les configurations visitées dans Λ sont de la forme $(p_i,u_iu\bot)$, où u_i est non vide, alors :

– $\tau_{u^{-1}}(\Lambda)$ est une partie partielle dans $\mathbb{G}(R)$ commençant en $(p,a\bot)$, où Eve respecte φ_R, et dans laquelle la pile n'est jamais vidée.
– si Λ se prolonge en une partie où le a est dépilé, alors la configuration ainsi atteinte est de la forme $(r,u\bot)$ avec $r \in R$.

Dès lors, si Eve respecte φ, $\varphi(\Lambda)$ est toujours défini. Afin de montrer que la stratégie φ est gagnante, considérons une partie Λ respectant φ. La partie Λ peut être de deux formes :

– dans Λ, toutes les configurations visitées sont de la forme $(p_i,u_iu\bot)$, où u_i est non vide. La partie Λ est donc l'image par τ_u d'une partie Λ' dans $\mathbb{G}(R)$ où la pile n'est jamais vidée et où Eve respecte φ_R. Ainsi, Λ' est dans $\Omega(R)$ et donc dans Ω (car la pile n'est jamais vidée). Dès lors, par invariance par translation vers le haut, Λ est également dans Ω
– $\Lambda = \Lambda' \cdot \Lambda_r$, où Λ_r est une partie dans $\mathbb{G}$, commençant par $(r,u\bot)$ pour un certain $r \in R$. De plus, Eve respecte φ_r dans Λ_r, ce qui implique que Λ_r est dans Ω. Dès lors, par invariance de Ω par translation vers la droite, on en conclut que Λ est gagnante pour Eve.

La stratégie φ est donc bien gagnante pour Eve dans $\mathbb{G}$ depuis $(p,au\bot)$.

Implication directe : Soit φ une stratégie gagnante pour Eve dans $\mathbb{G}$ depuis $(p,au\bot)$. Considérons l'ensemble $\mathcal{L}_\varphi$ des parties possibles dans $\mathbb{G}$ depuis $(p,au\bot)$, où Eve respecte φ. On définit alors un sous-ensemble R de Q de la façon suivante : pour tout état $r \in Q$, $r \in R$ s'il existe une partie Λ de $\mathcal{L}_\varphi$ de la forme $\Lambda' \cdot (r,u\bot) \cdot \Lambda''$, et où toutes les configurations dans Λ' sont de la forme $(p_i,v_iu\bot)$, où v_i est un mot non vide. Ainsi, R est l'ensemble des états de contrôle pouvant être atteints juste après avoir dépilé a dans une partie où Eve respecte φ.

Montrons que R est un ensemble de retours pour (p,a). Pour cela, on définit une stratégie φ_R pour Eve dans $\mathbb{G}(R)$ depuis $(p,a\bot)$. Pour toute partie partielle Λ dans $\mathbb{G}(R)$ depuis $(p,a\bot)$, on définit $\varphi_R(\Lambda)$ comme suit :

– si dans Λ aucune configuration de pile vide n'est visitée, $\varphi_R(\Lambda) = \tau_{u^{-1}}(\varphi(\tau_u(\Lambda)))$.
– si dans Λ une configuration de pile vide est visitée, $\varphi_R(\Lambda)$ peut prendre n'importe quelle valeur parmi celles qui sont possibles.

On a alors le résultat suivant : pour toute partie partielle Λ dans $\mathbb{G}(R)$ commençant en $(p,a\bot)$ où Eve respecte φ_R, on a :

1. si dans Λ la pile n'est jamais vidée, $\tau_u(\Lambda)$ est une partie partielle dans $\mathbb{G}$ commençant en $(p,au\bot)$, où Eve respecte φ.

2. si dans Λ une configuration de pile vide est visitée, alors $\Lambda = \Lambda' \cdot (r,\bot) \cdot \Lambda''$, où Λ' est une partie partielle dans laquelle la pile n'est jamais vide, et où $r \in R$.

Pour établir le résultat précédent, on raisonne par récurrence sur Λ :

– au départ, $\Lambda = (p,a\bot)$. La propriété est donc établie.

– soit maintenant une partie partielle Λ, pour laquelle on suppose le résultat établi. Soit Λ' un prolongement de Λ par un coup (d'Eve ou d'Adam).

Si Λ contenait déjà une configuration de pile vide, la seconde propriété étant vraie pour Λ, elle l'est encore pour Λ'.

Si Λ ne visite pas de configuration de pile vide, et si Λ' ne se termine pas par une configuration de pile vide, le premier point ne pose pas de problème : si Eve joue, cela vient de la définition de φ_R. Si c'est Adam qui joue, cela vient du fait que la dernière position de Λ et la dernière position de $\tau_u(\Lambda)$ ont le même état de contrôle et le même sommet de pile (les mêmes types de transitions peuvent donc être effectuées).

Enfin, si Λ ne visite pas de configuration de pile vide, et si Λ' se termine par une configuration de pile vide, en raisonnant comme ci-dessus, on établit que $\tau_u(\Lambda')$ est une partie partielle dans $\mathbb{G}$ commençant en $(p,au\bot)$, où Eve respecte φ. De plus, $\tau_u(\Lambda)$ est préfixe d'un élément de $\mathcal{L}_\varphi$. Par définition, l'état de contrôle dans la dernière position de $\tau_u(\Lambda')$ (qui prolonge $\tau_u(\Lambda)$) est donc dans R. Or, l'état de contrôle dans la dernière position de Λ' est le même que dans la dernière position de $\tau_u(\Lambda')$, ce qui prouve le second point.

Ainsi, une partie Λ dans $\mathbb{G}(R)$, où Eve respecte φ_R, est gagnante pour Eve. En effet, si la pile est vidée, l'état de contrôle dans la première configuration de pile vide visitée est un élément de R. Si la pile n'est jamais vidée, $\tau_u(\Lambda)$ est une partie de $\mathbb{G}$ commençant en $(p,au\bot)$, où Eve respecte φ. C'est donc une partie gagnante pour Eve. Or, $\Lambda = \tau_{u^{-1}}(\tau_u(\Lambda))$, et comme Ω est invariante par translation vers le bas, on en déduit que Λ est également dans Ω, et donc dans $\Omega(R)$ (puisque toutes les configurations visitées sont de pile non vide dans Λ).

L'ensemble R est donc bien un ensemble de retours pour (p,a) dans $\mathbb{G}$.

Il nous reste donc à établir que, pour tout $r \in R$, $(r,u\bot)$ est une position gagnante pour Eve dans $\mathbb{G}$.

Soit donc $r \in R$. Par définition de R, il existe une partie partielle, où Eve respecte φ, et qui est de la forme $\Lambda' \cdot (r,u\bot)$, où Λ' ne visite pas de configuration de pile vide. On définit une stratégie gagnante φ_r pour Eve dans $\mathbb{G}$ depuis $(r,u\bot)$ de la façon suivante : pour toute partie partielle Λ commençant dans $(r,u\bot)$, on pose $\varphi_r(\Lambda) = \varphi(\Lambda' \cdot \Lambda)$. Ainsi, pour toute partie Λ jouée selon φ_r, $\Lambda' \cdot \Lambda$ est une partie jouée selon φ et commençant dans $(p,au\bot)$. Dès lors, $\Lambda' \cdot \Lambda$ est une partie gagnante pour Eve, et par invariance de Ω par translation vers la gauche, Λ est également une partie gagnante. Dès lors $(r,u\bot)$ est gagnante pour Eve dans $\mathbb{G}$. □

Dans la remarque 4.1, nous avions noté que les ensembles d'ensembles de retours $\mathcal{R}(p,a)$ pouvaient être uniquement représentés par leur éléments minimaux pour l'inclusion. On peut, dans le cadre du lemme 4.3, se restreindre aux ensembles de retours minimaux. On établit sans difficulté le corollaire suivant.

Corollaire 4.1 *Soit p un état de contrôle de Q et soit a une lettre de Γ différente de $\bot$. Soit u un mot sur l'alphabet $(\Gamma \setminus \{\bot\})$, $(p,au\bot)$ est une position gagnante pour*

Eve dans le jeu $\mathbb{G}$, si et seulement s'il existe $R \in \mathcal{R}(p,a)$, minimal pour l'inclusion, tel que $(r,u\bot)$ est gagnant pour Eve dans $\mathbb{G}$ pour tout $r \in R$.

4.2.2 Régularité des ensembles de positions gagnantes

Considérons une configuration $(p,u\bot)$ dans le jeu $\mathbb{G}$ et posons $u = a_1a_2\cdots a_n$. En utilisant le lemme 4.3, on en déduit que $(p,u\bot)$ est gagnant pour Eve si et seulement s'il existe $R \in \mathcal{R}(p,a_1)$ tel que pour tout $r \in R$, (r,u') soit gagnant pour Eve, où l'on pose $u' = a_2a_3\cdots a_n$. Ainsi, on doit deviner un ensemble de retours R pour (p,a_1) et vérifier en parallèle pour tout $r \in R$, que les configurations (r,u') sont gagnantes, ce que l'on fait à nouveau en devinant des ensembles de retours, et ainsi de suite jusqu'à devoir vérifier que des configurations de la forme $(s,\bot)$ sont gagnantes. En d'autres termes, on a donc une procédure de décision qui lit le mot de pile de haut en bas (ou de gauche à droite), en alternant des actions existentielles (choix des ensembles de retours) avec des actions universelles (calculs en parallèle pour tous les états d'un ensemble de retours). Il s'agit ni plus ni moins d'un calcul typique d'un automate alternant.

On définit alors un $\mathcal{P}$-automate alternant $\mathcal{A} = \langle Q,\Gamma,\delta,F\rangle$ où :

- les états de $\mathcal{A}$ sont exactement ceux de $\mathcal{P}$.
- l'alphabet d'entrée est Γ, l'alphabet de pile de $\mathcal{P}$.
- la fonction de transition δ est définie par :

$$\delta(p,a) = \bigvee_{R\in\mathcal{R}(p,a)} \bigwedge_{r\in R} r$$

  si $a \neq \bot$.

  En particulier, si $\emptyset \in \mathcal{R}(p,a)$, $\delta(p,a) = tt$ et l'on retrouve alors le résultat du lemme 4.1. Symétriquement, si $\mathcal{R}(p,a) = \emptyset$, $\delta(p,a) = ff$ et l'on retrouve alors le résultat du lemme 4.2.

  Enfin, $\delta(p,\bot) = tt$ si Eve a une stratégie gagnante dans $\mathbb{G}$ depuis $(p,\bot)$, et sinon, $\delta(p,\bot) = ff$.
- l'ensemble des états finaux F est l'ensemble vide.

On a alors le résultat suivant.

Théorème 4.1 *Le $\mathcal{P}$-automate alternant $\mathcal{A}$ reconnaît l'ensemble des positions gagnantes pour Eve dans le jeu $\mathbb{G}$.*

Preuve. Montrons dans un premier temps que toute position gagnante pour Eve est acceptée par $\mathcal{A}$. On raisonne par récurrence sur la longueur du mot de pile. Soit donc une configuration $(p,u\bot)$ gagnante pour Eve dans $\mathbb{G}$.

- si $|u| = 0$, la configuration est donc de la forme $(p,\bot)$ et elle est acceptée par définition de $\delta(p,\bot)$.
- supposons le résultat établi pour toutes les configurations gagnantes de mot de pile, de longueur inférieure ou égale à un certain entier $n \geq 1$. Considérons une configuration gagnante pour Eve de longueur égale à $n+1$. Une telle configuration est de la forme $(p,au\bot)$, avec $|u| = n-1$. Comme $(p,au\bot)$ est

gagnante pour Eve, il existe, d'après le lemme 4.3, un ensemble de retours R pour (p,a) tel que $(r,u\bot)$ est gagnant pour tout $r \in R$. Or, par hypothèse de récurrence, on sait que $(r,u\bot)$ est accepté par $\mathcal{A}$ pour tout $r \in R$. Dès lors, le calcul de $\mathcal{A}$, qui consiste à choisir la transition initiale $\bigwedge_{r\in R} r$, et ensuite de *brancher* avec les calculs acceptant de $\mathcal{A}$ sur les $(r,u\bot)$, est un calcul acceptant.

Réciproquement, montrons que tout configuration acceptée par $\mathcal{A}$ est gagnante pour Eve dans le jeu $\mathbb{G}$. On raisonne à nouveau par récurrence sur la longueur du mot de pile. Soit donc une configuration $(p,u\bot)$ acceptée par $\mathcal{A}$.

- si $|u| = 0$, la configuration est donc de la forme $(p,\bot)$ et elle est gagnante par définition de $\delta(p,\bot)$.
- supposons le résultat établi pour toutes les configurations gagnantes de mot de pile de longueur inférieure ou égale à un certain entier $n \geq 1$. Considérons une configuration de longueur égale à $n+1$ et acceptée par $\mathcal{A}$. Une telle configuration est de la forme $(p,au\bot)$, avec $|u| = n-1$. Par définition de δ, il existe un ensemble de retours R pour (p,a) tel que, pour tout $r \in R$, $(r,u\bot)$ est accepté par $\mathcal{A}$. La configuration $(r,u\bot)$ est donc gagnante pour Eve par hypothèse de récurrence. Dès lors, d'après le lemme 4.3, $(p,au\bot)$ est une position gagnante pour Eve dans $\mathbb{G}$.

□

Remarque 4.2 Dans la définition de $\mathcal{A}$, les états finaux n'ont pas vraiment d'importance si l'on considère les calculs de cet automate sur des mots de pile valides, c'est-à-dire terminés par la lettre $\bot$. Ainsi, toute branche d'un calcul se termine par une transition étiqueté par $\bot$ et conduit à la formule tt ou ff. Dès lors, on peut faire n'importe quel choix pour F.

L'automate $\mathcal{A}$ est facilement complémentable, puisqu'il s'agit d'un automate alternant (l'automate complémentaire s'obtenant par dualisation). A nouveau, comme on l'a expliqué dans la remarque 4.2, les états finaux n'ont pas d'importance dans ce cas. On fait à nouveau le choix de prendre l'ensemble vide pour ensemble d'états finaux.

On définit donc le $\mathcal{P}$-automate alternant dual $\overline{\mathcal{A}} = \langle Q,\Gamma,\overline{\delta},F\rangle$, où :

- les états de $\overline{\mathcal{A}}$ sont exactement ceux de $\mathcal{P}$.
- l'alphabet d'entrée est Γ, l'alphabet de pile de $\mathcal{P}$.
- la fonction de transition $\overline{\delta}$ est définie par :

$$\overline{\delta}(p,a) = \bigwedge_{R\in\mathcal{R}(p,a)} \bigvee_{r\in R} r$$

si $a \neq \bot$.

En particulier, si $\emptyset \in \mathcal{R}(p,a)$, $\overline{\delta}(p,a) = ff$ et l'on retrouve le résultat du lemme 4.1 Si $\mathcal{R}(p,a) = \emptyset$, $\overline{\delta}(p,a) = tt$ et l'on retrouve le résultat du lemme 4.2.

Enfin, $\overline{\delta}(p,\bot) = tt$, si Adam a une stratégie gagnante dans $\mathbb{G}$ depuis $(p,\bot)$, et sinon, $\overline{\delta}(p,\bot) = ff$.

– l'ensemble des états finaux est l'ensemble vide.

On a alors le Corollaire suivant du théorème 4.1

Corollaire 4.2 *Le $\mathcal{P}$-automate alternant $\overline{\mathcal{A}}$ reconnaît l'ensemble des positions gagnantes pour Adam dans le jeu $\mathbb{G}$.*

Remarque 4.3 Dans les constructions des automates $\mathcal{A}$ et $\overline{\mathcal{A}}$, on considère tous les ensembles de retours. En fait, en raison du corollaire 4.1, les automates construits en remplaçant les ensembles $\mathcal{R}(p,a)$ par les ensembles d'ensembles de retours minimaux, reconnaissent encore les ensembles de positions gagnantes.

En conclusion, on a le résultat suivant.

Théorème 4.2 *Soit $\mathbb{G} = (\mathcal{G},\Omega)$ un jeu sur un graphe de jeu engendré par un processus à pile. Si la condition de gain Ω est borélienne et invariante par translations verticales et horizontales, alors les ensembles de positions gagnantes sont réguliers.*

Remarque 4.4 Dans l'énoncé précédent, la condition Ω borélienne sert à assurer que les ensembles de retours sont bien définis. On peut donc la remplacer par la condition plus faible suivante : $\forall R \subseteq Q$, $\mathbb{G}(R)$ est déterminé.

Le théorème 4.2 implique en particulier la régularité des ensembles de positions gagnantes pour des jeux sur des graphes de processus à pile, munis d'une condition de Büchi, de parité, de Muller, de bornage ou de toute combinaison booléenne de telles conditions.

Exemple 4.4 Considérons le jeu introduit dans l'exemple 4.3, et déterminons le $\mathcal{P}$-automate alternant $\mathcal{A} = \langle Q,\Gamma,\delta,F\rangle$, acceptant l'ensemble des positions gagnantes pour Eve, donné par le théorème 4.1. Dans l'exemple 4.3, nous avons déjà déterminé les ensembles de retours minimaux, ce qui permet de définir δ pour les lettres différentes de $\bot$. Pour définir $\delta(p,\bot)$, $\delta(q,\bot)$ et $\delta(r,\bot)$, il nous faut déterminer si les configurations $(p,\bot)$, $(q,\bot)$ et $(r,\bot)$ sont gagnantes pour Eve dans $\mathbb{G}$.

– la figure 4.11 décrit la structure de $\mathcal{G}$ autour de $(p,\bot)$. La flèche en pointillé allant de $(r,b\bot)$ à $(p,\bot)$ signifie qu'Eve peut forcer à revenir depuis $(r,b\bot)$ à $(p,\bot)$ (Cf. figures 4.6 et 4.10). Eve peut donc forcer une boucle sur $(p,\bot)$. La plus petite couleur apparaissant dans une telle boucle étant 0, on en déduit alors que $(p,\bot)$ est une position gagnante pour Eve dans $\mathbb{G}$.

FIG. 4.11 – *Structure de $\mathcal{G}$ autour de $(p,\bot)$*

– la figure 4.12 décrit la structure de $\mathcal{G}$ autour de $(q,\bot)$. Il est clair que $(q,\bot)$ est une position gagnante pour Eve dans $\mathbb{G}$, puisque Eve peut atteindre la position gagnante $(p,\bot)$ depuis $(q,\bot)$.

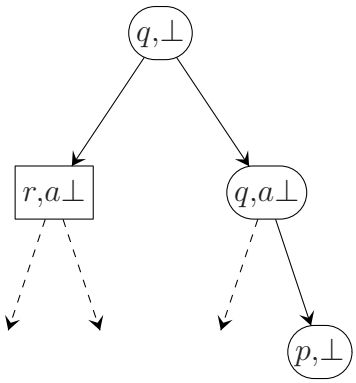


FIG. 4.12 – *Structure de $\mathcal{G}$ autour de $(q,\bot)$*

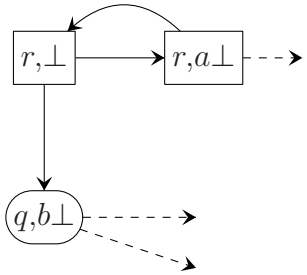


FIG. 4.13 – *Structure de $\mathcal{G}$ autour de $(r,\bot)$*

– la figure 4.13 décrit la structure de $\mathcal{G}$ autour de $(r,\bot)$. Il est clair que $(r,\bot)$ est une position gagnante pour Adam dans $\mathbb{G}$, puisque Adam peut boucler sur $(r,\bot)$ via $(r,a\bot)$, et donc répéter infiniment la couleur 1.

On a donc les ensembles de retours suivants (restreints à leurs éléments minimaux) :

$$\begin{array}{lcllcl} \mathcal{R}(p,a) & = & \{\{q\},\{p,r\}\} & \mathcal{R}(p,b) & = & \{\{p\}\} \\ \mathcal{R}(q,a) & = & \{\{p\}\} & \mathcal{R}(q,b) & = & \{\{p\}\} \\ \mathcal{R}(r,a) & = & \{\{p,r\}\} & \mathcal{R}(r,b) & = & \{\{p\}\} \end{array}$$

Et les résultats suivants concernant les configurations de pile vide :

$$(p,\bot) \in W_{\mathbf{E}} \qquad (q,\bot) \in W_{\mathbf{E}} \qquad (r,\bot) \in W_A$$

On a donc la fonction de transition suivante :

$$\begin{array}{lcllcllcl} \delta(p,a) & = & q \vee (p \wedge r) & \delta(p,b) & = & p & \delta(p,\bot) & = & tt \\ \delta(q,a) & = & p & \delta(q,b) & = & p & \delta(q,\bot) & = & tt \\ \delta(r,a) & = & p \wedge r & \delta(r,b) & = & p & \delta(r,\bot) & = & ff \end{array}$$

On peut alors facilement déterminer l'ensemble des configurations acceptées par $\mathcal{A}$, c'est-à-dire l'ensemble des configurations gagnantes pour Eve :

$$W_{\mathbf{E}} = L(\mathcal{A}) = (Q \times (\Gamma \setminus \{\bot\})^*\bot) \setminus (\{r\} \times (a^*\bot))$$

On a également les configurations gagnantes pour Adam :

$$W_{\mathbf{A}} = \{r\} \times (a^*\bot)$$

4.2.3 A propos de l'effectivité

La construction du $\mathcal{P}$-automate précédemment décrite n'est pas effective dans le cas général. Le problème réside dans le calcul des ensembles de retours $\mathcal{R}(p,a)$. Le fait que Ω soit borélienne assure que ces derniers sont bien définis, mais n'implique nullement leur effectivité.

Pour s'en convaincre un peu plus, donnons dès maintenant un exemple de jeu $\mathbb{G} = (\mathcal{G},\Omega)$, joué sur un graphe de processus à pile équipé d'une condition de gain invariante par translations et pour lequel on ne peut pas calculer les ensembles de retours. Une première possibilité serait de définir Ω de façon non constructive, en disant par exemple que Ω est l'ensemble de toutes les parties, si une propriété P est vraie, et que sinon Ω est vide. La condition Ω est bien invariant par translations mais décider le gagnant (ou calculer les ensembles de retours) est équivalent à décider P. Il suffit alors de choisir pour P une propriété indécidable.

On peut cependant reprocher à l'argument précédent de considérer une condition de gain totalement ineffective, puisque l'on ne peut proposer de modèle de machine décidant si une partie est dans Ω. Eve et Adam ne pouvant tricher, essayons d'en faire autant en proposant un contre-exemple plus constructif. Considérons donc une machine de Turing M déterministe et acceptant des mots finis sur un alphabet Q. Supposons de plus que l'on puisse décider si un mot u est accepté par M, mais que l'on ne puisse décider le vide pour le langage $L(M)$ reconnu par cette même machine. On pourra par exemple considérer que M simule l'intersection de deux automates à pile déterministes $\mathcal{A}_1$ et $\mathcal{A}_2$: décider l'appartenance d'un mot à $L(M) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ ne pose pas de problème, tandis que décider si $L(M)$ est vide est indécidable pour un bon choix de $\mathcal{A}_1$ et $\mathcal{A}_2$. Considérons maintenant que Ω est l'ensemble des parties pour lesquelles la suite des états de contrôles, considérée comme un mot, est infinie et contient des facteurs, commençant à partir de positions arbitrairement éloignées, dans $L(M)$. Plus formellement :

$$\Omega = \{(p_1,u_1)(p_2,u_2)(p_3,u_3)\cdots \mid p_1p_2p_3\cdots \in (Q^*L(M))^\omega\}$$

On montre facilement que Ω est invariant par translations horizontales et verticales. De plus, il n'est pas difficile de proposer une machine de Turing non déterministe équipée d'une condition de Büchi et qui reconnaisse (en un sens à définir) Ω, ce qui rend Ω un peu plus effectif que dans l'exemple précédent.

Considérons maintenant le graphe G engendré par le processus à pile $\mathcal{P} = \langle Q,\{a,\bot\},\bot,\Delta\rangle$, où l'on a $\Delta(p,b) = \{skip(q) \mid q \in Q\}$ pour tout $p \in Q$ et toute lettre $b \in \{a,\bot\}$. Considérons maintenant le graphe de jeu $\mathcal{G}$ induit par la partition suivante de Q : $Q_{\mathbf{E}} = Q$ et $Q_{\mathbf{A}} = \emptyset$. Il est alors clair que si Ω est non vide, Eve gagne depuis n'importe quelle position et réciproquement. Cependant savoir si Ω est vide est indécidable, car c'est un problème équivalent à décider le vide de $L(M)$.

Une grande partie de cette thèse est consacrée à proposer des méthodes de calcul des ensembles de retours pour diverses conditions de gain, et ainsi obtenir l'effectivité de la construction précédente, afin de décrire les ensembles des positions gagnantes

4.2.4 Comment décider le gagnant depuis une position quelconque?

Supposons que la construction précédente soit effective et que l'on possède donc un $\mathcal{P}$-automate alternant $\mathcal{A} = \langle Q,\Gamma,\delta,F\rangle$ qui reconnaît l'ensemble des positions gagnantes pour Eve. Etant donnée une configuration $(p,u\bot)$, comment décider, de façon efficace, si Eve possède une stratégie gagnante dans $\mathbb{G}$ depuis $(p,u\bot)$?

L'idée est de lire le mot $u\bot$ de droite à gauche (c'est-à-dire de bas en haut en terme de pile), et de maintenir un ensemble d'états de $\mathcal{A}$. Soit n, la longueur de $u\bot$. Au départ, ce sous-ensemble est initialisé à $Q_n = \emptyset$. Lorsque l'on lit une lettre a et qu'on a l'ensemble Q_i, Q_{i-1} est égal à l'ensemble des états q tels que $\delta(q,a)$ est satisfaite par Q_i (c'est-à-dire par la valuation où tous les éléments de Q_i sont mis à 1 et tous les autres à 0). On note alors $Q_i \models \delta(q,a)$). En particulier, Q_{n-1} est l'ensemble des états f, tels que $(f,\bot)$ est gagnant pour Eve dans $\mathbb{G}$. La configuration $(p,u\bot)$ est alors gagnante pour Eve, si et seulement si $p \in Q_0$. L'algorithme 1 présente cette procédure.

algorithme 1 Décide l'acceptation par $\mathcal{A}$ d'une configuration (p,u)

Entrée: Un mot $u = u_1u_2\cdots u_n$, un état p.
Sortie: (p,u) est-il accepté par $\mathcal{A}$?

```
n ← |u|
Q_n ← F = ∅
Pour i = n − 1 à 0 Faire
  a ← u[i + 1]
  Pour tout q ∈ Q Faire
    Si Q_{i+1} ⊨ δ(q,a) alors
      Ajouter q dans Q_i
    Fin du Si
  Fin du Pour
Fin du Pour
Si p ∈ Q_0 alors
  Retourne "(p,u) est accepté"
Sinon
  Retourne "(p,u) est rejeté"
Fin du Si
```

Nous allons prouver la validité de cet algorithme. En effet, bien que celle-ci soit connue [24], la preuve nous sera utile par la suite.

Proposition 4.3 *Soit $(p,u) \in Q \times ((\Gamma \setminus \{\bot\})^*\bot)$. La configuration (p,u) est acceptée par $\mathcal{A}$ si et seulement si l'algorithme 1 répond que (p,u) est acceptée. De plus, l'algorithme 1 fonctionne en temps linéaire en la longueur de u et en espace linéaire en la taille de Q.*

Preuve. On raisonne par récurrence sur la longueur de u :

- si $|u| = 1$, le résultat est immédiat : la configuration est acceptée par l'algorithme 1 si et seulement si $\emptyset \models \delta(p,\bot)$, c'est-à-dire si et seulement si $\delta(p,\bot) = tt$, c'est-à-dire si et seulement si elle est acceptée par l'automate $\mathcal{A}$.
- supposons le résultat établi pour des mots u de longueur $n \geq 1$ et montrons qu'il est encore vrai pour des mots de longueurs $n+1$. Soit u de longueur $n+1$: u est alors de la forme $u = av$, où v est un mot de longueur n.
  Supposons que (p,u) soit accepté par $\mathcal{A}$. Il existe alors un calcul acceptant de $\mathcal{A}$ sur (p,u). Dans ce calcul, considérons l'ensemble $R \in \mathcal{R}(p,a)$ choisi dans la première transition. L'automate $\mathcal{A}$ accepte donc tous les (r,v) pour $r \in R$ (un calcul acceptant est donné par le sous-arbre du calcul acceptant de $\mathcal{A}$ sur (p,u), de racine r et de hauteur n). Par hypothèse de récurrence, l'algorithme 1 accepte donc toutes les entrées (r,v) pour $r \in R$. Dans le déroulement de l'algorithme sur l'entrée (p,u), on a donc $R \subseteq Q_1$. On en conclut alors que $p \in Q_0$ et donc que l'algorithme 1 accepte également (p,u).
  Réciproquement, si l'algorithme 1 accepte (p,u), dans le déroulement de ce dernier sur l'entrée (p,u), on a $p \in Q_0$. Dès lors, $Q_1 \models \delta(p,a) = \bigvee_{R \in \mathcal{R}(p,a)} \bigwedge_{r \in R} r$.
  Il existe donc $R \in \mathcal{R}(p,a)$ tel que $R \subseteq Q_1$. Il s'ensuit que (r,v) est accepté par l'algorithme 1 pour tout $r \in R$ et, par hypothèse de récurrence, que $\mathcal{A}$ accepte (r,v) pour tout $r \in R$. Considérons le calcul de $\mathcal{A}$ sur (p,u), obtenu en choisissant les éléments de R pour successeurs de p, et en branchant ensuite les calculs acceptants respectifs de $\mathcal{A}$ sur chacun des (r,v) pour $r \in R$. On obtient alors un calcul acceptant de $\mathcal{A}$ sur (p,u), ce qui conclut la preuve.

La complexité ne pose pas de problème. □

Exemple 4.5 En reprenant le jeu de l'exemple 4.3, sur l'entrée $(r,aba\bot)$, l'algorithme 1 accepte et calcule la suite : $Q_4 = \emptyset$, $Q_3 = \{p,q\}$, $Q_2 = \{p,q\}$, $Q_1 = \{p,q,r\}$ et $Q_0 = \{p,q,r\}$ qui contient r.

En fait, la preuve ci-dessus nous donne une méthode permettant, pour toute configuration (p,u) acceptée par $\mathcal{A}$, de construire un arbre de calcul acceptant. C'est ce que décrit l'algorithme 2.

La validité de l'algorithme 2 est une conséquence directe de celle de l'algorithme 1. Tout comme l'algorithme 1, et à condition de ne pas faire plusieurs fois le même appel récursif, l'algorithme 2 fonctionne en temps linéaire en la longueur de u et en espace linéaire en la taille de Q.

Exemple 4.6 Dans le cadre de l'exemple 4.3, sur l'entrée $(r,aba\bot)$, et sur la suite $Q_4 = \emptyset$, $Q_3 = \{p,q\}$, $Q_2 = \{p,q\}$, $Q_1 = \{p,q,r\}$ et $Q_0 = \{p,q,r\}$, l'algorithme 2 donne, par exemple, l'arbre décrit dans la figure 4.14.

4.2.5 Composition des stratégies

Dans ce qui suit, on suppose à nouveau que la construction des ensembles de retours est effective mais aussi que l'on possède des stratégies gagnantes effectives

algorithme 2 Donne un calcul acceptant de $\mathcal{A}$ à l'aide de l'algorithme 1

Entrée: Un mot $u = u_1u_2\cdots u_n$, un état p. La suite $Q_0,\ldots,Q_n$ calculée dans l'algorithme 1 sur (p,u)

Sortie: Donne un arbre de calcul acceptant de $\mathcal{A}$ sur (p,u).

$n \leftarrow |u|$

Si $n > 0$ **alors**

Choisir $R \in \mathcal{R}(p,u_1)$ tel que $R \subseteq Q_1$

Pour tout $r \in R$ **Faire**

Calculer récursivement l'arbre T_r associé à $(r,u_2\cdots u_n)$ et $Q_1,\ldots,Q_n$

Fin du Pour

Retourner l'arbre de racine p et de fils $(T_r)_{r\in R}$

Sinon

Retourner l'arbre de racine p et d'unique feuille $\#$

Fin du Si

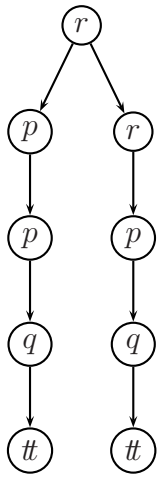


FIG. 4.14 – *Exemple 4.6 : arbre de calcul acceptant de* $\mathcal{A}$ *sur* $(r,aba\bot)$.

pour Eve dans les jeux conditionnels $\mathbb{G}(R)$ depuis les positions $(p,a\bot)$, pour tout $p \in Q$, tout $a \in \Gamma \setminus \{\bot\}$ et tout $R \in \mathcal{R}(p,a)$. On suppose en outre que l'on a des stratégies gagnantes effectives pour Eve dans $\mathbb{G}$ depuis les positions $(p,\bot)$ gagnantes pour Eve.

L'objet de ce qui suit est de montrer comment déduire une stratégie gagnante pour Eve dans $\mathbb{G}$ depuis toute position gagnante. La construction repose sur les algorithmes 1 et 2, ainsi que sur la preuve du Lemme 4.3.

Soit une configuration de la forme $(p,au\bot)$ gagnante pour Eve dans le jeu $\mathbb{G}$. D'après le lemme 4.3, il existe un ensemble de retours $R \in \mathcal{R}(p,a)$ tel que $(r,u\bot)$ est une position gagnante pour Eve dans $\mathbb{G}$ pour tout $r \in R$ (et l'on désigne par φ_r une stratégie gagnante associée). De plus, la preuve nous fournissait une stratégie gagnante φ pour Eve depuis $(p,au\bot)$, construite à partir d'une stratégie φ_R gagnante pour Eve depuis $(p,a\bot)$ dans $\mathbb{G}(R)$. Soit $\Lambda = v_0v_1v_2\cdots v_n$ une partie partielle. On

définit $\varphi(\Lambda)$ de la façon suivante :

- si toute configuration v_i apparaissant dans Λ est de la forme $(p_i,u_iu\bot)$, où u_i est non vide, on pose $\varphi(\Lambda) = \tau_u(\varphi_R(\tau_{u^{-1}}(\Lambda)))$.
- s'il existe un entier j tel que $v_j = (r,u\bot)$ pour un certain $r \in R$, et tel que pour tout $i < j$, v_i est de la forme $(p_i,u_iu\bot)$, on pose $\varphi(\Lambda) = \varphi_r(v_jv_{j+1}\cdots v_n)$.
- sinon $\varphi(\Lambda)$ n'est pas défini.

La stratégie φ n'est pas définissable en l'état actuel des choses, puisque nous n'avons pas défini les stratégies φ_r (sauf dans le cas où u est vide, auquel cas les stratégies sont connues d'après les hypothèses faites ci-dessus). On procède donc par récurrence et l'on obtient l'algorithme 3.

La validité de l'algorithme 3 vient de la preuve du lemme 4.3 et des invariants suivants, facilement démontrables :

- h est égal à la plus petite hauteur de pile visitée depuis le début de la partie.
- a est la première lettre de u à n'avoir jamais été dépilée ($a = u_{n-h+1}$).
- av est le plus long suffixe de u qui a toujours été présent en fond de pile depuis le début de la partie.
- q était l'état de contrôle lorsque la première configuration de hauteur h a été atteinte.
- $R \in \mathcal{R}(q,a)$, et pour tout $r \in R$, (r,v) est gagnante pour Eve dans $\mathbb{G}$.
- si $h > 1$, $\tau_{v^{-1}}(\Lambda)$ est un préfixe de partie dans $\mathbb{G}(R)$ commençant en $(q,a\bot)$, où Eve respecte sa stratégie gagnante $\varphi_{q,a,R}$.
- si $h = 1$, Λ est un préfixe de partie dans $\mathbb{G}$ commençant en $(q,\bot)$, où Eve respecte sa stratégie gagnante $\varphi_{q,\bot}$.

La stratégie ainsi décrite utilise, en plus de la puissance de calcul des stratégies $\varphi_{q,a,R}$ et $\varphi_{q,\bot}$, un arbre fini de hauteur n, où n est la longueur du mot de pile dans la position initiale. Ainsi, seule une mémoire finie additionnelle est nécessaire. Par ailleurs, un précalcul, celui de l'arbre de calcul du $\mathcal{P}$-automate alternant $\mathcal{A}$ reconnaissant les positions gagnantes est nécessaire. Comme précisé précédemment, ce calcul demande un temps linéaire dans la taille de la configuration initiale et un espace linéaire dans le nombre d'états du processus à pile dont $\mathbb{G}$ est issu.

Exemple 4.7 Dans le cadre des exemples 4.3 et 4.6, décrivons une stratégie gagnante pour Eve, donnée par l'algorithme 3, dans le jeu précédemment introduit depuis la position $(r,aba\bot)$, pour laquelle un arbre de calcul acceptant de $\mathcal{A}$ a été donné dans l'exemple 4.6. Nous ne décrivons ici que les stratégies $\varphi_{q,a,R}$ et $\varphi_{q,\bot}$ utiles. Chacune de ces stratégies, que nous choisissons positionnelles, ne sera décrite que sur les configurations effectivement visitables. Cela donne :

- $\varphi_{r,a,\{p,r\}}$: définie sur aucune configuration (c'est Adam qui joue depuis $(r,a\bot)$ et il ne peut que dépiler).
- $\varphi_{p,b,\{p\}}$: $\varphi_{p,b,\{p\}}(p,b\bot) = (p,\bot)$.
- $\varphi_{r,b,\{p\}}$: $\varphi_{r,b,\{p\}}(q,ab\bot) = (p,b\bot)$ et $\varphi_{r,b,\{p\}}(p,b\bot) = (p,\bot)$.
- $\varphi_{p,a,\{q\}}$: $\varphi_{p,a,\{q\}}(p,a\bot) = (q,\bot)$.

algorithme 3 Stratégie gagnante pour Eve dans $\mathbb{G}$

Entrée: Un mot $u = u_1u_2\cdots u_n$, un état p tels que (p,u) soit une position gagnante pour Eve dans $\mathbb{G}$.

Entrée: Des stratégies gagnantes pour Eve $\varphi_{q,a,R}$ dans $\mathbb{G}(R)$ depuis $(q,a\bot)$ pour tout $R \in \mathcal{R}(q,a)$, pour tout $q \in Q$ et tout $a \in \Gamma \setminus \{\bot\}$. Des stratégies gagnantes $\varphi_{q,\bot}$ pour toute position $(q,\bot)$ gagnante pour Eve dans $\mathbb{G}$

Sortie: Une stratégie gagnante pour Eve dans $\mathbb{G}$ depuis (p,u).

$n \leftarrow |u|$
$h \leftarrow n$
$a \leftarrow u_1$
$v \leftarrow u_2\cdots u_n$
$q = p$
$\Lambda = (p,u)$
$T \leftarrow$ Arbre de calcul acceptant de $\mathcal{A}$ sur (p,u)
$R \leftarrow$ Etiquettes des fils de la racine de T
Boucle
  Si $h > 1$ **alors**
    Tant que Plus petite configuration visitée est de hauteur h **Faire**
      Si position actuelle appartient à Eve **alors**
        Jouer $\tau_v(\varphi_{q,a,R}(\tau_{v^{-1}}(\Lambda)))$
      Fin du Si
      $\Lambda \leftarrow (\Lambda \cdot$ nouvelle position$)$
    Fin du Tant que
  Sinon
    Boucle
      Si position actuelle appartient à Eve **alors**
        Jouer $\varphi_{q,\bot}(\Lambda)$
      Fin du Si
      $\Lambda \leftarrow (\Lambda \cdot$ nouvelle position$)$
    Fin du Boucle
  Fin du Si
  $h \leftarrow h - 1$
  $a \leftarrow u_{n-h+1}$
  $v \leftarrow u_{n-h+2}\cdots u_n$
  $q \leftarrow$ état de contrôle de la dernière position de Λ
  $\Lambda \leftarrow$ position courante
  $T \leftarrow$ sous-arbre de T de hauteur h et de racine d'étiquette q
  $R \leftarrow$ étiquette des fils de la racine de T
Fin du Boucle

– $\varphi_{q,\perp}$: $\varphi_{q,\perp}(q,\perp) = (q,a\perp)$, $\varphi_{q,\perp}(q,a\perp) = (p,\perp)$, $\varphi_{q,\perp}(p,\perp) = (r,b\perp)$, $\varphi_{q,\perp}(q,ab\perp) = (p,b\perp)$ et $\varphi_{q,\perp}(p,b\perp) = (p,\perp)$.

La stratégie φ donnée par l'algorithme 3 sur $(r,aba\perp)$ vide la pile jusqu'à atteindre la configuration $(q,\perp)$ et ensuite joue selon $\varphi_{q,\perp}$. On ne décrit φ que sur les configurations effectivement visitables, et on la décompose selon les valeurs prises par h dans l'algorithme :

– $h = 4$. Eve ne joue pas dans ce cas. En fait Adam depuis $(r,aba\perp)$ va en $(p,ba\perp)$ ou en $(r,ba\perp)$.
– $h = 3$. On a deux possibilités :
  – on est allé depuis $(r,aba\perp)$ en $(p,ba\perp)$. On a alors $R = \{p\}$, et φ est alors construite à partir de $\varphi_{p,b,\{p\}}$. On a alors $\varphi(p,ba\perp) = (p,a\perp)$.
  – on est allé depuis $(r,aba\perp)$ en $(r,ba\perp)$. On a alors $R = \{p\}$, et φ est alors construite à partir de $\varphi_{r,b,\{p\}}$. On a alors $\varphi(q,aba\perp) = (p,ba\perp)$, et $\varphi(p,ba\perp) = (p,a\perp)$.
– $h = 2$. La stratégie φ est dans tous les cas construite à partir de $\varphi_{p,a,\{q\}}$. On a alors $\varphi(p,a\perp) = (q,\perp)$
– $h = 1$. On pose $\varphi = \varphi_{q,\perp}$.

Dans ce cas, la stratégie φ construite est positionnelle et elle est donnée par :

$$\begin{array}{lll}
\varphi(p,ba\perp) = (p,a\perp) & \varphi(q,aba\perp) = (p,ba\perp) & \varphi(p,a\perp) = (q,\perp) \\
\varphi(q,\perp) = (q,a\perp) & \varphi(q,a\perp) = (p,\perp) & \varphi(p,\perp) = (r,b\perp) \\
\varphi(q,ab\perp) = (p,b\perp) & \varphi(p,b\perp) = (p,\perp) &
\end{array}$$

4.3 Généralisation

Comme nous l'avons noté dans l'exemple 4.2, les conditions d'accessibilité ne sont pas invariantes par translations vers la gauche. Pour ces conditions, qui sont parmi les plus naturelles, les résultats du paragraphe 4.2 ne peuvent donc pas être appliqués. En particulier, on ne peut pas utiliser le théorème 4.2 et conclure que les ensembles de positions gagnantes sont des ensembles réguliers.

Il en va de même, comme nous l'avons vu dans l'exemple 4.1, pour les conditions de type accessibilité, Büchi, parité ou Muller, lorsque les configurations à répéter sont caractérisées non plus par leurs états de contrôle, mais par leurs contenus de pile.

L'objet de ce paragraphe est de montrer que, pour certaines conditions, on peut se ramener au cadre du paragraphe 4.2, et ainsi prouver la régularité des ensembles de positions gagnantes pour une grande famille de conditions de gain.

Dans le paragraphe 4.3.1, on s'intéressera aux conditions d'accessibilité et aux conditions ω-régulières sur les états dont les conditions d'accessibilité sont un cas particulier. Ensuite, dans le paragraphe 4.3.2, on proposera un résultat beaucoup plus général qui utilise la notion d'enrichissement déterministe. Ce résultat implique en fait ceux du paragraphe 4.3.1. On peut donc, si on le souhaite, sauter le paragraphe 4.3.1 et n'en retenir que le résultat principal, à savoir le Corollaire 4.4. Cependant les constructions du paragraphe 4.3.1 permettent probablement de mieux

comprendre et de rendre plus concrète la notion d'enrichissement de jeu. Enfin, dans les paragraphes 4.3.3 et 4.3.4, on donne deux applications des résultats du paragraphe 4.3.2, dont le théorème 4.4 qui peut être vu comme l'aboutissement de ce chapitre, puisqu'il propose une classe générale de conditions de gain pour lesquelles les positions gagnantes sont des ensembles réguliers.

4.3.1 Conditions d'accessibilité et conditions ω-régulières sur les états

On commence tout d'abord par se donner un graphe de jeu $\mathcal{G} = (V,E)$ engendré par un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$.

Dans un premier temps, on considère une condition d'accessibilité Ω qui demande que l'on atteigne une configuration dont l'état de contrôle appartient à un sous-ensemble d'états de contrôle que l'on note F. Ainsi, $\Omega = V^*(F \times \Gamma^*)V^\omega$. On note $\mathbb{G} = (\mathcal{G},\Omega)$.

Un jeu d'accessibilité peut être vu comme un jeu faisant intervenir un arbitre qui, dès qu'une configuration finale a été visitée, lèverait un drapeau (et le maintiendrait levé), signifiant ainsi que la partie est remportée par Eve. Dans cette optique, et du fait que l'arbitre garde le drapeau en l'air une fois celui-ci levé, la condition de gain est une condition de Büchi : Eve gagne si et seulement si l'arbitre lève infiniment souvent son drapeau. Concrètement, on va construire un graphe $\mathcal{G}'$ obtenu en enrichissant $\mathcal{G}$, et l'on va considérer un jeu de Büchi $\mathbb{G}'$ sur $\mathcal{G}'$, tel que les ensembles de positions gagnantes dans $\mathbb{G}$ soient obtenus par projection à partir des positions gagnantes dans $\mathbb{G}'$ (qui forment un ensemble régulier d'après le théorème 4.2, en tant que jeu de Büchi). La régularité des positions gagnantes dans $\mathbb{G}$ vient de la régularité des positions gagnantes dans $\mathbb{G}'$, et de la préservation de la régularité par projection.

Plus formellement, on considère le processus à pile $\mathcal{P}' = \langle Q',\Gamma,\bot,\Delta'\rangle$ défini par :

- $Q' = Q \times \{0,1\}$, la seconde composante simule le drapeau de l'arbitre;
- la définition de Δ' est basée sur celle de Δ :
  - $push((q,1),b) \in \Delta((p,1),a)$ si et seulement si $push(q,b) \in \Delta(p,a)$;
  - $pop((q,1)) \in \Delta((p,1),a)$ si et seulement si $pop(q) \in \Delta(p,a)$;
  - $skip((q,1)) \in \Delta((p,1),a)$ si et seulement si $skip(q) \in \Delta(p,a)$;
  - $push((q,0),b) \in \Delta((p,0),a)$ si et seulement si $push(q,b) \in \Delta(p,a)$ et $q \notin F$;
  - $pop((q,0)) \in \Delta((p,0),a))$ si et seulement si $pop(q) \in \Delta(p,a)$ et $q \notin F$;
  - $skip((q,0)) \in \Delta((p,0),a))$ si et seulement si $skip(q) \in \Delta(p,a)$ et $q \notin F$;
  - $push((q,1),b) \in \Delta((p,0),a)$ si et seulement si $push(q,b) \in \Delta(p,a)$ et $q \in F$;
  - $pop((q,1)) \in \Delta((p,0),a)$ si et seulement si $pop(q) \in \Delta(p,a)$ et $q \in F$;
  - $skip((q,1)) \in \Delta((p,0),a)$ si et seulement si $skip(q) \in \Delta(p,a)$ et $q \in F$.

On note $\mathcal{G}' = (V',E')$ le graphe de processus à pile engendré par $\mathcal{P}'$, et l'on pose $F' = Q \times \{1\}$ le sous-ensemble d'états de Q' de seconde composante égale à 1. Enfin, on considère Ω', la condition de Büchi sur F', et l'on considère $\mathbb{G}' = (\mathcal{G}',\Omega')$, le jeu de Büchi sur $\mathcal{G}'$.

On a alors le résultat suivant.

Proposition 4.4 *Soit p un état de contrôle de Q, et soit u un mot de pile valide sur l'alphabet Γ. La configuration (p,u) est gagnante pour Eve dans le jeu $\mathbb{G}$ si et seulement si la configuration $((p,i),u)$ est gagnante pour Eve dans le jeu $\mathbb{G}'$, où $i = 0$ si $p \notin F$ et $i = 1$ sinon.*

Preuve. Le cas où $p \in F$ est trivial. On suppose donc que $p \notin F$.

Supposons dans un premier temps que la configuration (p,u) soit gagnante pour Eve dans le jeu $\mathbb{G}$, et considérons une stratégie gagnante φ pour Eve. Comme $\mathbb{G}$ est muni d'une condition d'accessibilité, on peut choisir φ positionnelle, ce qui permettra d'alléger les notations. On définit une stratégie positionnelle φ' pour Eve dans $\mathbb{G}'$ en posant $\varphi'((q,i),v) = ((r,j),w)$, où $(r,w) = \varphi(q,v)$ et où $j = 1$ si et seulement $i = 1$ ou $r \in F$. La stratégie φ' imite φ tout en mettant à jour la seconde composante des états de contrôle dans le graphe enrichi $\mathcal{G}'$.

Soit τ la projection de Q' sur Q définie par $\tau(q,i) = q$ pour tout $q \in Q$ et pour tout $i \in \{0,1\}$. La projection τ s'étend naturellement en une projection de V' sur V, et plus généralement en un morphisme lettre à lettre des parties dans $\mathbb{G}'$ sur les parties dans $\mathbb{G}$.

On montre facilement que, pour toute partie Λ' de $\mathbb{G}'$ depuis $((p,0),u)$, où Eve respecte φ', la partie $\Lambda = \tau(\Lambda')$ obtenue par projection à partir de Λ' est une partie de $\mathbb{G}$ depuis (p,u), où Eve respecte φ. Ainsi, Λ visite une configuration finale et donc Λ' passe par une configuration finale et visite infiniment souvent des configurations finales (puisqu'une fois qu'une configuration finale a été atteinte, on ne visite plus que des configurations finales). Ainsi, Λ' est gagnante pour Eve.

Réciproquement, supposons qu'Eve possède une stratégie gagnante dans $\mathbb{G}'$ depuis $((p,0),u)$. Considérons donc une stratégie positionnelle, gagnante φ' pour Eve dans $\mathbb{G}'$ depuis $((p,0),u)$. On définit alors une stratégie φ pour Eve dans $\mathbb{G}$ par $\varphi(\Lambda) = (r,w)$, avec $((r,j),w) = \varphi'((q,i),v)$, où (q,v) est la dernière position de Λ, et où $i = 1$ si et seulement si Λ contient une configuration finale.

Comme précédemment, pour toute partie Λ où Eve respecte φ, la partie Λ', obtenue en enrichissant les états de contrôle des configurations de Λ par un bit d'information codant la visite à un état final dans le début de partie, est une partie de $\mathbb{G}'$, où Eve respecte φ' et est donc gagnante, ce qui implique que Λ est remportée par Eve. □

De la proposition 4.4, on en déduit immédiatement le corollaire suivant.

Corollaire 4.3 *Soit $\mathbb{G}$ un jeu sur un graphe de jeu engendré par un processus à pile et muni d'une condition d'accessibilité sur les états. Alors les ensembles de positions gagnantes sont réguliers.*

Plus généralement, on peut s'intéresser aux conditions de gain suivantes, que l'on appelle *conditions de gain ω-régulières sur les états*

Définition 4.7 *Soit Ω une condition de gain. On dit que Ω est une condition de gain* ω-régulière sur les états *si et seulement s'il existe un langage ω-régulier L tel que $\Omega = \{(p_1,u_1)(p_2,u_2)\cdots \mid p_1p_2\cdots \in L\}$. En d'autres termes, les conditions ω-régulières sur les états sont celles pour lesquelles l'issue d'une partie est déterminée par l'appartenance de la suite des états visités à un langage ω-régulier.*

Exemple 4.8 Toute les conditions d'accessibilité, de Büchi, de parité ou de Muller, lorsqu'elles portent sur les états de contrôles, sont des conditions de gain ω-régulières sur les états.

Cependant les conditions de gain ω-régulières sont plus générales que les conditions de Muller. En effet, étant donné un jeu sur un graphe de processus à pile engendré à partir d'un processus à pile à deux états notés p et q, la condition de gain demandant que la suite des états appartienne au langage $\{p,q\}^*(pq)^\omega$ est une condition de gain ω-régulière sur les états. On voit facilement qu'elle ne peut être exprimée comme une condition de Muller portant sur les états de contrôles.

Comme les conditions d'accessibilité sont des conditions de gain ω-régulières, ces dernières ne sont pas invariantes par translations horizontales. Cependant, on peut généraliser les idées de la proposition 4.4, et conclure à la régularité des ensembles de positions gagnantes.

Dans le cas des conditions d'accessibilité, on utilisait un arbitre pour détecter la visite d'un état final. Dans cette construction, deux points sont importants : le comportement de l'arbitre est déterministe et une fois le drapeau levé, celui-ci le reste. Le comportement de l'arbitre peut être résumé par l'automate déterministe de Büchi $\mathcal{A}$ donné par la figure 4.15.

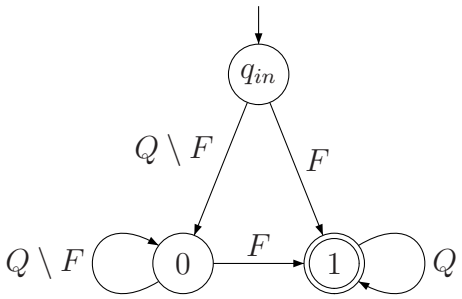


FIG. 4.15 – *Arbitre pour la condition d'accessibilité*

Cet automate traduit en effet comment la position du drapeau évolue en fonction de sa position actuelle : baissée (0) ou levée (1). Cet automate est en fait l'automate reconnaissant le langage ω-régulier Q^*FQ^ω, associé à la condition (ω-régulière) d'accessibilité dans le cadre de la définition 4.7. Le processus à pile $\mathcal{P}'$ était obtenu à partir de $\mathcal{P}$ en l'enrichissant des états de $\mathcal{A}$, et en faisant fonctionner celui-ci à la volée. Dans la proposition 4.4, pour décider si une configuration (p,u) était gagnante dans $\mathbb{G}$, on considérait la configuration $((p,i),u)$, où i est l'état atteint depuis l'état initial de $\mathcal{A}$ en lisant p. Etant donné un jeu sur un processus à pile et une condition de gain ω-régulière sur les états Ω, on va donc considérer un automate déterministe $\mathcal{A}$ reconnaissant le langage régulier sous-jacent à Ω, en faire le *produit* avec le processus à pile et considérer le jeu associé au nouveau processus à pile afin de décider le gagnant dans le jeu originel. Les automates déterministes de Büchi ne reconnaissant pas tous les langages ω-réguliers, on va choisir pour $\mathcal{A}$ un automate déterministe muni d'une condition de parité.

Plus formellement, supposons qu'on ait un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ et une condition de gain ω-régulière sur les états dont le langage ω-régulier sous-jacent est un langage L sur l'alphabet Q. On s'intéresse au jeu $\mathbb{G} = (\mathcal{G},\Omega)$, où $\mathcal{G}$ est un graphe engendré par $\mathcal{P}$ et une partition de $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de Q. On considère un automate déterministe $\mathcal{A} = (T,Q,\{i\},\rho,\delta)$ muni d'une condition de parité $\rho : Q \to C$ dans un ensemble C de couleurs, et tel que $L(\mathcal{A}) = L$. On construit alors un processus à pile $\mathcal{P}' = \langle Q',\Gamma,\bot,\Delta'\rangle$ de la façon suivante :

- $Q' = Q \times T$.
- la définition de Δ' est basée sur celle de Δ :
  - on a $push((p',t'),b) \in \Delta((p,t),a)$ si et seulement si $push(p',b) \in \Delta(p,a)$ et $\delta(t,p') = t'$.
  - on a $pop((p',t')) \in \Delta((p,t),a)$ si et seulement si $pop(p') \in \Delta(p,a)$ et $\delta(t,p') = t'$.
  - on a $skip((p',t')) \in \Delta((p,t),a)$ si et seulement si $skip(p') \in \Delta(p,a)$ et $\delta(t,p') = t'$.

On note $\mathcal{G}' = (V',E')$ le graphe de processus à pile engendré par $\mathcal{P}'$ et par la partition $(Q_{\mathbf{E}} \times T) \cup (Q_{\mathbf{A}} \times T)$ de Q'. Enfin, on étend ρ en une fonction de Q' dans C en posant $\rho'(q,t) = \rho(t)$, et l'on considère le jeu $\mathbb{G}' = (\mathcal{G}',\Omega')$, où Ω' est la condition de parité induite par ρ'.

On a alors le résultat suivant généralisant la proposition 4.4

Proposition 4.5 *Soit p un état de contrôle de Q, et soit u un mot de pile valide sur l'alphabet Γ. La configuration (p,u) est gagnante pour Eve dans le jeu $\mathbb{G}$ si et seulement si la configuration $((p,j),u)$ est gagnante pour Eve dans le jeu $\mathbb{G}'$, où $j = \delta(i,p)$.*

Preuve. Soit τ la projection naturelle de Q' sur Q définie par $\tau(q,t) = q$ pour tout $q \in Q$, et pour tout $t \in T$. La projection τ s'étend naturellement en une projection de V' sur V, et en un morphisme lettre à lettre des parties dans $\mathbb{G}'$ sur les parties dans $\mathbb{G}$.

Soit τ_1 la projection naturelle de Q' sur Q définie par $\tau_1(q,t) = q$ pour tout $q \in Q$, et pour tout $t \in T$. La projection τ_1 s'étend naturellement en une projection de V' sur Q et enfin en un morphisme lettre à lettre des parties dans $\mathbb{G}'$ sur Q^∞. De même, on définit la projection naturelle τ_2 de Q' sur T par $\tau_2(q,t) = t$ pour tout $q \in Q$ et pour tout $t \in T$. La projection τ_2 s'étend naturellement en une projection de V' sur T et enfin, en un morphisme lettre à lettre des parties de $\mathbb{G}'$ sur T^∞.

Par construction de $\mathcal{P}'$, on a facilement que pour toute partie Λ' dans $\mathbb{G}'$ depuis $((p,j),u)$, $i \cdot \tau_2(\Lambda')$ est la suite des états de $\mathcal{A}$ obtenue en lisant le mot $\tau_1(\Lambda)$ depuis i.

Supposons dans un premier temps que la configuration (p,u) soit gagnante pour Eve dans le jeu $\mathbb{G}$, et appelons φ une stratégie gagnante pour Eve depuis (p,u). On va définir une stratégie φ' pour Eve dans $\mathbb{G}'$, qui imite φ tout en mettant à jour la seconde composante des états de contrôle dans le graphe enrichi $\mathcal{G}'$. Pour cela, on pose, pour tout préfixe de partie Λ' dans $\mathbb{G}'$ commençant en $((p,j),u)$, $\varphi'(\Lambda') = ((r,\delta(t,r)),w)$, où la dernière position de Λ' est de la forme $((q,t),v)$ et où $(r,w) = \varphi(\tau(\Lambda'))$.

Soit Λ' une partie jouée selon φ' et commençant en $((p,j),u)$. Considérons la partie $\Lambda = \tau(\Lambda')$ obtenue par projection de Λ'. De la définition de φ', on conclut facilement que Λ est une partie de $\mathbb{G}$ depuis (p,u), où Eve respecte la stratégie φ. Dès lors, Λ est acceptée par $\mathcal{A}$ et donc $i \cdot \tau_2(\Lambda')$ est un calcul acceptant de $\mathcal{A}$. Les couleurs infiniment souvent visitées dans Λ' étant les mêmes que dans $i \cdot \tau_2(\Lambda')$, on en conclut que Λ' est une partie gagnante pour Eve. Ainsi, $((p,j),u)$ est une position gagnante pour Eve dans $\mathbb{G}'$.

Réciproquement, supposons que $((p,j),u)$ soit une position gagnante pour Eve dans $\mathbb{G}'$, et appelons φ' une stratégie gagnante pour Eve depuis $((p,j),u)$. On va définir, à partir de φ', une stratégie φ pour Eve dans $\mathbb{G}$. Considérons une partie partielle Λ dans $\mathbb{G}$ commençant en (p,u), appelons ν la suite des états dans Λ, et μ la suite des états de $\mathcal{A}$, privée de son état initial i, obtenue en lisant ν. Ainsi, Λ et μ sont de même longueur. Appelons Λ' la suite obtenu en appliquant à Λ et μ le morphisme lettre à lettre défini de $V^\infty \times T^\infty$ dans V'^∞ par $((q,v),t) \mapsto ((q,t),v)$. La suite Λ' est donc obtenue en enrichissant Λ avec le calcul de $\mathcal{A}$ sur sa suite d'états.

La stratégie φ est alors définie par $\varphi(\Lambda) = (q,v)$ où $((q,t),v) = \varphi'(\Lambda')$. Ainsi, φ ajoute de l'information à Λ pour pouvoir utiliser φ', et enfin supprime l'information superflue pour être à valeur dans V. On montre alors facilement que, pour toute partie Λ de $\mathbb{G}$ commençant en (p,u), où Eve respecte φ, Λ' est une partie de $\mathbb{G}'$ commençant en $((p,j),u)$ où Eve respecte φ. La partie Λ' est donc gagnante pour Eve et la suite $i \cdot \tau_2(\Lambda')$ correspond donc à un calcul acceptant de $\mathcal{A}$. Or $i \cdot \tau_2(\Lambda')$ est le calcul de $\mathcal{A}$ sur la suite des états de Λ. On en conclut donc que Λ est accepté par $\mathcal{A}$, et donc que Λ est une partie gagnante pour Eve. Ainsi, (p,u) est une position gagnante pour Eve dans $\mathbb{G}$. □

De la proposition 4.5, on en déduit immédiatement le corollaire suivant.

Corollaire 4.4 *Soit $\mathbb{G}$ un jeu sur un graphe de jeu engendré par un processus à pile et muni d'une condition de gain ω-régulière sur les états. Alors les ensembles de positions gagnantes sont réguliers.*

Remarque 4.5 Nous l'avons déjà évoqué, décider le gagnant dans un jeu sur un graphe de processus à pile muni d'une condition de parité, peut être réalisé en temps exponentiel dans le nombre de couleurs et dans le nombre d'états du processus à pile sous-jacent au jeu (voir [83, 84] ainsi que le chapitre 5). La proposition 4.5 fournit donc une procédure effective pour décider le gagnant dans un jeu sur un graphe de processus à pile muni d'une condition de gain ω-régulière sur les états. Cette procédure est donc exponentielle dans la taille du processus à pile, mais aussi dans la taille de l'automate $\mathcal{A}$, c'est-à-dire dans son nombre d'états et dans le nombre de couleurs qu'il utilise.

Ainsi, une objection naturelle à la construction proposée est la suivante : pourquoi utiliser un automate déterministe avec une condition de parité alors qu'on pourrait utiliser un automate de Büchi non déterministe, qui posséderait ainsi moins d'états et n'emploierait que deux couleurs, pour coder Ω dans le graphe $\mathcal{G}'$?

Choisir un automate non déterministe pour représenter le langage L sous-jacent à Ω et en faire un *produit* avec $\mathcal{G}$ reviendrait à considérer un jeu $\mathbb{G}'$ sur un graphe de processus à pile $\mathcal{G}'$, où Eve devrait deviner ($\mathcal{A}$ n'est plus déterministe), en même temps qu'elle joue, un calcul acceptant de $\mathcal{A}$ sur la suite des états visités dans la

partie. Cette contrainte supplémentaire ne préserve pas les ensembles de positions gagnantes dans le sens de la proposition 4.5, comme le montre le contre-exemple qui suit.

Considérons le langage régulier $L = p_0p(\{q_1^\omega\} \cup \{q_2^\omega\})$, sur l'alphabet $Q = \{p_0,p,q_1,q_2\}$, reconnu par l'automate non déterministe de Büchi donné par la figure 4.16.

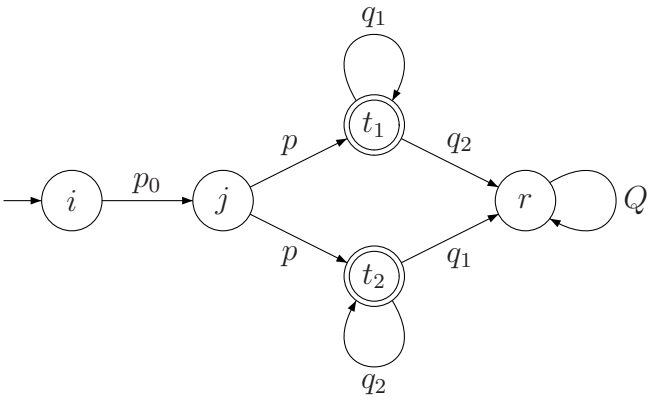


FIG. 4.16 – *Automate non déterministe acceptant* $L = p_0p(\{q_1^\omega\} \cup \{q_2^\omega\})$

Imaginons maintenant un jeu très simple sur un graphe de processus à pile d'états Q, et où $Q_{\mathbf{E}}$ est vide (c'est donc Adam qui va effectuer tous les coups). Imaginons en plus qu'il y a seulement deux parties possibles à partir de la configuration $(p_0,\bot)$: une dont la suite des états est $p_0pq_1^\omega$ et une autre dont la suite des états est $p_0pq_2^\omega$. La position $(p_0,\bot)$ est donc gagnante pour Eve. Dans le jeu obtenu en prenant le produit avec $\mathcal{A}$, ces deux suites d'états vont donner quatre suites d'états possibles depuis $((p_0,j),\bot)$ selon le choix effectué à la lecture de l'état p : $(p_0,j),(p,t_1)(q_1,t_1)^\omega$ et $(p_0,j),(p,t_2)(q_2,t_2)^\omega$, qui sont gagnantes pour Eve, mais aussi $(p_0,j),(p,t_1)(q_2,r)^\omega$ et $(p_0,j),(p,t_2)(q_1,r)^\omega$ qui sont perdantes pour Eve. Or, une fois le choix de l'état de $\mathcal{A}$, obtenu en lisant p depuis j effectué, Adam peut prolonger la partie en une partie perdante pour Eve (en allant dans l'état q_2 si Eve a choisi t_1 et en allant dans l'état q_1 si Eve a choisi t_2). Ainsi, $((p_0,j),\bot)$ est gagnante pour Adam.

Plus formellement, un tel jeu est facilement obtenu en considérant le processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ où :

- $Q = \{p_0,p,q_1,q_2\}$
- $\Gamma = \{\bot\}$.
- $\Delta(p_0,\bot) = \{skip(p)\}$.
- $\Delta(p,\bot) = \{skip(q_1),skip(q_2)\}$.
- $\Delta(q_1,\bot) = \{skip(q_1)\}$.
- $\Delta(q_2,\bot) = \{skip(q_2)\}$.

De plus, on peut remarquer que dans ce cas, le graphe de jeu $\mathcal{G}$ est fini. Le résultat est donc également vrai dans le cas des jeux sur des graphes finis.

4.3.2 Enrichissement déterministe de jeu

On souhaite généraliser les constructions du paragraphe précédent. La clef de voûte de celles-ci résidait dans la construction d'un jeu sur un graphe de processus à pile, obtenu à partir du jeu initial en lui ajoutant de l'information qui est calculée de façon déterministe. On considère la formalisation suivante de cette idée :

Définition 4.8 *Soit $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ un processus à pile et soit $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ une partition de Q. On note $\mathcal{G} = (V,E)$ le graphe de jeu associé, et l'on se donne une condition de gain Ω sur $\mathcal{G}$. Enfin, on pose $\mathbb{G} = (\mathcal{G},\Omega)$.*

Soient Q_m un nouvel ensemble d'états, Γ_m un nouvel ensemble de symboles de pile, et $\bot_m \in \Gamma_m$ un fond de pile. On pose alors

- $Q' = Q \times Q_m$.
- $\Gamma' = [(\Gamma \setminus \{\bot\}) \times (\Gamma_m \setminus \{\bot_m\})] \cup \{\bot'\}$.
- $\bot' = (\bot,\bot_m)$.

On considère alors un processus à pile $\mathcal{P}' = \langle Q',\Gamma',\bot',\Delta'\rangle$, où Δ' possède les propriétés suivantes :

- *pour toute règle $push(p',b) \in \Delta(p,a)$ avec $a \neq \bot$, pour tout état $t \in Q_m$ et toute lettre $\alpha \in \Gamma_m$, il existe un unique état $t' \in Q_m$ et une unique lettre $\beta \in \Gamma_m$ tels que $push((p',t'),(b,\beta)) \in \Delta'((p,t),(a,\alpha))$;*
- *pour toute règle $skip(p') \in \Delta(p,a)$ avec $a \neq \bot$, pour tout état $t \in Q_m$ et toute lettre $\alpha \in \Gamma_m$, il existe un unique état $t' \in Q_m$, tel que $skip((p',t')) \in \Delta'((p,t),(a,\alpha))$;*
- *pour toute règle $pop(p') \in \Delta(p,a)$, pour tout état $t \in Q_m$ et toute lettre $\alpha \in \Gamma_m$, il existe un unique état $t' \in Q_m$, tel que $pop((p',t')) \in \Delta'((p,t),(a,\alpha))$;*
- *pour toute règle $push(p',b) \in \Delta(p,\bot)$, pour tout état $t \in Q_m$, il existe un unique état $t' \in Q_m$ et une unique lettre $\beta \in \Gamma_m$, tels que $push((p',t'),(b,\beta)) \in \Delta'((p,t),\bot')$;*
- *pour toute règle $skip(p') \in \Delta(p,\bot)$, pour tout état $t \in Q_m$, il existe un unique état $t' \in Q_m$, tel que $skip((p',t')) \in \Delta'((p,t),\bot')$.*

On dit alors que $\mathcal{P}'$ est un enrichissement déterministe *de $\mathcal{P}$.*

Soit la partition $Q'_{\mathbf{E}} \cup Q'_{\mathbf{A}}$ de Q' donnée par $Q'_{\mathbf{E}} = Q_{\mathbf{E}} \times Q_m$ et $Q'_{\mathbf{A}} = Q_{\mathbf{A}} \times Q_m$. On appelle $\mathcal{G}' = (V',E')$ le graphe engendré par $\mathcal{P}'$ et la précédente partition. Enfin, on considère une condition de gain Ω' sur $\mathcal{G}'$, et l'on note $\mathbb{G}' = (\mathcal{G}',\Omega')$ le jeu associé.

On appelle τ la projection de V' sur V définie par $\tau((p,t),(a,\alpha)) = (p,a)$ et étendue en un morphisme lettre à lettre de V'^{∞} sur V^{∞}.

Soit f une fonction de V^{∞} dans V'^{∞} telle que :

- *f préserve les longueurs : $|f(\Lambda)| = |\Lambda|$ pour tout $\Lambda \in V^{\infty}$.*
- *pour tout $\Lambda \in V^{+}$ et pour tout sommet $v \in V$, $f(\Lambda \cdot v) = f(\Lambda) \cdot v'$ pour un certain $v' \in V'$.*
- *f est* conforme *à E et E' : pour tout $\Lambda \in V^{+}$ se terminant dans un sommet v_1, pour tout $v_2 \in V$ tel que $(v_1,v_2) \in E$: $f(\Lambda \cdot v_2) = f(\Lambda) \cdot v'_2$ avec $(v'_1,v'_2) \in E'$, où v'_1 est le dernier sommet de $f(\Lambda)$.*
- *f est* conforme *avec τ : pour tout $\Lambda \in V^{\infty}$, $\tau(f(\Lambda)) = \Lambda$.*

Enfin, on dit que le jeu $\mathbb{G}'$ est un enrichissement déterministe *du jeu $\mathbb{G}$ par f, si pour tout $\Lambda \in V^\infty$, $\Lambda \in \Omega$ si et seulement si $f(\Lambda) \in \Omega'$.*

Exemple 4.9 Les constructions données dans le paragraphe 4.3.1 pour les conditions d'accessibilité et pour les conditions ω-régulières sont des enrichissements déterministes.

De la définition 4.8, on déduit les propriétés suivantes.

Propriété 4.2 *Soit un jeu $\mathbb{G}$, et soit $\mathbb{G}'$ un enrichissement déterministe de $\mathbb{G}$ par une fonction f. Avec les notations de la définition 4.8, on a les propriétés suivantes :*

1. *pour toute partie partielle Λ' dans $\mathbb{G}'$, $\tau(\Lambda')$ est une partie partielle dans $\mathbb{G}$.*
2. *pour toute partie partielle Λ dans $\mathbb{G}$, $f(\Lambda)$ est une partie partielle dans $\mathbb{G}'$.*
3. *soient (v,v_1),$(v,v_2) \in E'$, deux arcs dans $\mathcal{G}'$. Si $\tau(v_1) = \tau(v_2)$ alors $v_1 = v_2$.*
4. *soit $\Lambda = v_0v_1v_2\cdots$ une partie dans $\mathbb{G}$. Soit $\Lambda' = v'_0v'_1v'_2\cdots$ l'image de Λ par f. Alors pour tout $j \geq 1$, v'_j est uniquement déterminé par v_j et v'_{j-1}. Dès lors, $f(\Lambda)$ est uniquement déterminée par Λ et par la valeur de f sur le premier sommet de f.*
5. *pour toute configuration (p,u) dans $\mathcal{G}$ et pour toute partie Λ' dans $\mathbb{G}'$ depuis $f((p,u))$, $f(\tau(\Lambda')) = \Lambda'$.*

Preuve.

1. Ce point vient du fait que $\mathcal{P}'$ est un enrichissement déterministe de $\mathcal{P}$.
2. Le résultat vient du fait que f est conforme à E et E'.
3. Le résultat vient de la définition de Δ' à partir de Δ
4. Soit donc $j \geq 1$. Par conformité de f avec τ, on déduit que la projection de v'_j par τ est v_j et que celle de v'_{j-1} est v_{j-1}. Par ailleurs, par conformité de f avec E et E', il existe un arc de v'_{j-1} vers v'_j dans $\mathcal{G}'$. Dès lors, le troisième point nous permet de conclure.
5. $\tau(f(\tau(\Lambda'))) = \tau(\Lambda')$ par conformité de f avec τ. Dès lors, Λ' et $f(\tau(\Lambda'))$ ont la même image par τ et commencent toutes les deux par la même configuration, à savoir $f((p,u))$. Du quatrième point, on déduit qu'elles sont égales.

□

On a alors le résultat suivant.

Théorème 4.3 *Soit un jeu $\mathbb{G}$ et soit $\mathbb{G}'$ un enrichissement déterministe de $\mathbb{G}$ par une fonction f. Alors pour toute configuration (p,u) dans $\mathbb{G}$, (p,u) est gagnante pour Eve dans $\mathbb{G}$, si et seulement si $f((p,u))$ est gagnante dans $\mathbb{G}'$ pour Eve.*

De plus, on peut déduire de façon effective, à partir d'une stratégie dans $\mathbb{G}$ depuis (p,u), une stratégie dans $\mathbb{G}'$ depuis $f((p,u))$.

Preuve. Soit une configuration (p,u) dans $\mathbb{G}$, gagnante pour Eve. Appelons φ une stratégie gagnante pour Eve dans $\mathbb{G}$ depuis (p,u). On définit alors une stratégie gagnante φ' pour Eve dans $\mathbb{G}'$ depuis $f((p,u))$, qui imite φ. Pour cela, on pose, pour toute partie partielle Λ' dans $\mathbb{G}'$, $\Lambda = \tau(\Lambda')$ (où τ est la projection introduite dans la définition 4.8) et $v = \varphi(\Lambda)$. Enfin, on pose $\varphi'(\Lambda) = v'$, où v' est tel que $f(\Lambda \cdot v) = f(\Lambda) \cdot v'$.

Comme $\mathbb{G}'$ est un enrichissement déterministe de $\mathbb{G}$, de la définition de φ' on déduit que, pour toute partie Λ' depuis $f((p,u))$ dans $\mathbb{G}'$ où Eve respecte φ', $\tau(\Lambda')$ est une partie dans $\mathbb{G}$ depuis (p,u) où Eve respecte φ. Donc, pour toute partie Λ', dans $\mathbb{G}'$, depuis $f((p,u))$ et où Eve respecte φ', $\tau(\Lambda') \in \Omega$ car c'est une partie où Eve respecte φ. Dès lors, $\Lambda' = f(\tau(\Lambda')) \in \Omega$. D'où l'implication directe.

Réciproquement, supposons que $f((p,u))$ soit une position gagnante pour Eve dans $\mathbb{G}'$. Appelons φ' une stratégie gagnante pour Eve dans $\mathbb{G}'$ depuis $f((p,u))$. On définit alors une stratégie gagnante φ pour Eve dans $\mathbb{G}$ depuis (p,u), qui imite φ'. Pour cela, on pose, pour toute partie partielle Λ dans $\mathbb{G}$, $\Lambda' = f(\Lambda)$ et l'on pose $v = \varphi'(\Lambda')$. Enfin, on pose $\varphi(\Lambda) = \tau(v)$

Comme $\mathbb{G}'$ est un enrichissement déterministe de $\mathbb{G}$, de la définition de φ on déduit que pour toute partie Λ depuis (p,u) dans $\mathbb{G}$, où Eve respecte φ, $f(\Lambda)$ est une partie dans $\mathbb{G}'$ depuis $f((p,u))$, où Eve respecte φ'. Donc, pour toute partie Λ, dans $\mathbb{G}$, depuis (p,u), et où Eve respecte φ, $f(\Lambda) \in \Omega'$, car c'est une partie où Eve respecte φ'. Dès lors, $\Lambda \in \Omega$. D'où l'implication réciproque.

La seconde partie, qui concerne l'effectivité, est immédiate d'après ce qui précède. □

En ce qui concerne la régularité des ensembles de positions gagnantes, on introduit la définition suivante.

Définition 4.9 *Soit un jeu $\mathbb{G}$ issu d'un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$, et soit $\mathbb{G}'$ un enrichissement déterministe de $\mathbb{G}$ par une fonction f, issu d'un processus à pile $\mathcal{P}' = \langle Q',\Gamma',\bot,\Delta'\rangle$.*

Si f est telle que tout ensemble régulier de $Q' \times \Gamma'^*$ est transformé par image inverse en un ensemble régulier de $Q \times \Gamma^*$, alors on dit que $\mathbb{G}'$ est un enrichissement déterministe régulier *de $\mathbb{G}$ par f.*

Exemple 4.10 Tout enrichissement déterministe par une fonction réalisable par un transducteur est un enrichissement régulier.

On a alors le corollaire suivant du théorème 4.3

Corollaire 4.5 *Soit un jeu $\mathbb{G}$ et soit $\mathbb{G}'$ un enrichissement déterministe régulier de $\mathbb{G}$ par une fonction f. Si dans $\mathbb{G}'$ les ensembles de positions gagnantes sont réguliers, alors ils le sont également dans $\mathbb{G}$.*

Remarque 4.6 Les constructions concernant les conditions d'accessibilité, et les conditions de gain ω-régulières données dans le paragraphe 4.3.1 sont en fait des enrichissements déterministes réguliers, et les résultats obtenus sont alors des applications du corollaire 4.5.

4.3.3 Conditions régulières de pile

On donne maintenant une application du théorème 4.3 et du corollaire 4.5.

Considérons la condition de gain suivante : Eve gagne une partie si elle vide infiniment souvent la pile. Une telle condition de gain n'est pas invariante par translation vers le haut, et l'on ne peut dès lors pas conclure que l'ensemble des positions gagnantes est un langage régulier.

On s'intéresse maintenant aux conditions de gain suivantes, que l'on appelle *conditions régulières de pile*

Définition 4.10 *Soit un alphabet de pile* Γ*, et soit un ensemble d'états* Q*. Soit* $\mathcal{C}_1, \ldots, \mathcal{C}_n$ *des langages réguliers de mots finis sur l'alphabet* Γ*, formant une partition des mots de piles valides sur* Γ*, c'est-à-dire de* $(\Gamma \setminus \{\bot\})^*\bot$*. On définit la fonction de coloriage d'un mot de pile par* $col(u) = i$*, où* i *est tel que* $\widetilde{u} \in \mathcal{C}_i$ *(on lit donc* u *de bas en haut en terme de pile), que l'on étend sur les couples de* $Q \times (\Gamma \setminus \{\bot\})^*\bot$ *en posant* $col((p,u)) = col(u)$*. Enfin, on étend col en un morphisme lettre à lettre des parties dans un jeu sur un graphe de processus à pile d'états* Q *et d'alphabet* Γ*, dans* $\{1, \ldots, n\}^\infty$*. Soit* L *un langage* ω*-régulier sur l'alphabet des couleurs* $\{1, \ldots, n\}$*. On définit enfin une condition de gain* Ω*, que l'on qualifie de* condition régulière de pile*, par* $\Omega = \{\Lambda \mid col(\Lambda) \in L\}$*.*

Exemple 4.11 La condition demandant que des configurations de pile vide soient infiniment souvent visitées est une condition régulière de pile. En effet, il suffit de poser $\mathcal{C}_1 = \{\bot\}$ et $\mathcal{C}_2 = (\bot(\Gamma\setminus\{\bot\})^*)\setminus\mathcal{C}_1$, et enfin de considérer le langage ω-régulier $L = (2^*1)^\omega$.

Commençons par expliquer comment procéder pour la condition donnée dans l'exemple 4.11, où Ω spécifie que l'on visite infiniment souvent des configurations de pile vide. Pour cela, il suffit de coder dans les états quand une configuration de pile vide vient d'être visitée. On enrichit donc Q en $Q' = Q \times \{1,2\}$, et les transitions de la forme $push(q,a) \in \Delta(p,\bot)$ deviennent $push((q,1),a) \in \Delta((p,i),\bot)$ pour tout $i = 1,2$ et les transitions de la forme $skip(q) \in \Delta(p,\bot)$ deviennent $skip((q,1)) \in \Delta((p,i),\bot)$ pour tout $i = 1,2$. Les transitions effectuées depuis une configuration de pile non vide, c'est-à-dire de la forme $push(q,b)$,$pop(q)$,$skip(q) \in \Delta(p,a)$ deviennent respectivement $push((q,2),b)$,$pop((q,2))$,$skip((q,2)) \in \Delta((p,i),a)$ pour tout $i = 1,2$. La condition de gain dans le nouveau jeu est une condition de Büchi demandant de visiter infiniment souvent des configurations d'état de contrôle dans $Q \times \{1\}$. Ainsi, on est dans une configuration d'état de contrôle dans $Q \times \{1\}$ si et seulement si la configuration précédente était de pile vide, et dès lors on répète infiniment souvent une configuration de pile vide si et seulement si on visite infiniment souvent un état de $Q \times \{1\}$.

En terme d'automates sur les mots finis, les automates de la figure 4.17 reconnaissent respectivement les langages $\mathcal{C}_1$ et $\mathcal{C}_2$. Ces deux automates diffèrent seulement par leur état final et c'est leur comportement que l'on codait dans la seconde composante de Q'.

L'idée de la construction précédente réside dans le codage dans les états de contrôle de l'état de contrôle d'un automate déterministe acceptant le langage $\mathcal{C}_1$ (information qui ici code également l'appartenance à $\mathcal{C}_2$). Cette information sur le mot de pile a un temps de retard en ce sens qu'à tout moment, l'état code l'information relative à la pile dans la configuration précédente.

Lors d'une opération d'empilement, il est facile de mettre à jour cette information (il suffit de simuler la transition de l'automate depuis l'état courant à la lecture de la nouvelle lettre empilée). Lors d'une opération de type *skip*, il n'y a rien à mettre à jour. En revanche, lors d'un dépilement, l'automate codant le langage de pile n'étant

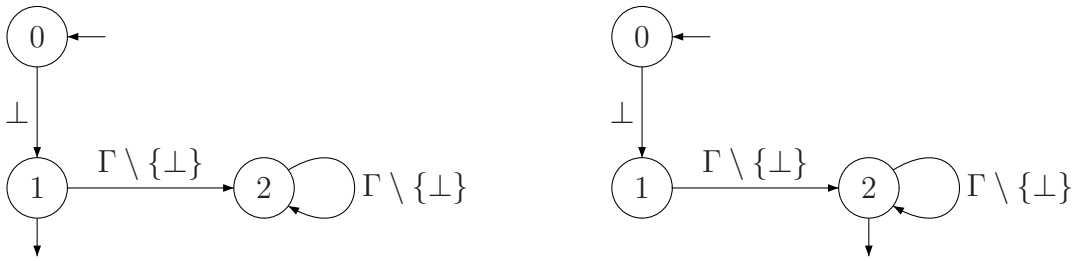


FIG. 4.17 – *Automates reconnaissant $\mathcal{C}_1$ et $\mathcal{C}_2$*

en général pas co-déterministe, on ne peut savoir dans quel état conduit le nouveau mot de pile. Dans le cas de l'exemple 4.11, la solution est simple : on regarde au coup d'après si la pile est vide.

Dans le cas général, la solution adoptée sera la suivante : on construit un nouveau processus à pile, dans lequel on code l'information relative aux états d'automates déterministes associés à chaque $\mathcal{C}_i$, dans les états de contrôle mais aussi dans la pile. Le sommet de pile contiendra toujours l'information relative à la configuration courante, tandis que le contrôle contiendra celle relative à la configuration précédente, information qu'il obtiendra à l'aide du sommet de pile. La mise à jour de l'information dans le sommet de pile n'est nécessaire que lors d'un empilement. Celle dans l'état de contrôle est obligatoire dans tous les cas et se fait en recopiant celle contenue dans le sommet de pile, d'où le temps de retard.

Plus formellement, on rappelle que l'on s'est donné un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$, une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de Q, ainsi qu'une condition régulière de pile Ω, de langages sous-jacents $\mathcal{C}_1,\ldots,\mathcal{C}_n$, et de langage ω-régulier associé L. On se fixe pour chaque langage $\mathcal{C}_j$ un automate déterministe $\mathcal{A}_j$ reconnaissant $\mathcal{C}_j$. On pose pour chaque $j = 1,\ldots,n$, $\mathcal{A}_j = \langle Q_j,\Gamma,\{i_j\},F_j,\delta_j\rangle$.

On définit maintenant un enrichissement régulier de $\mathbb{G}$ en respectant les notations de la définition 4.8. Pour cela, on définit $\mathcal{P}' = \langle Q',\Gamma',\bot',\Delta'\rangle$:

- $Q_m = (Q_1 \times Q_2 \times \cdots \times Q_n) \cup \{\#\}$.
- $\Gamma_m = (Q_1 \times Q_2 \times \cdots \times Q_n) \cup \{\bot_m\}$.
- $Q' = Q \times Q_m$.
- $\bot' = (\bot,\bot_m)$.
- $\Gamma' = [(\Gamma \setminus \{\bot\}) \times (\Gamma_m \setminus \{\bot_m\})] \cup \{\bot'\}$.

La définition de Δ' est basée sur celle de Δ et simule les automates $\mathcal{A}_1,\ldots,\mathcal{A}_n$:

- $push((p',(q'_1,\ldots,q'_n)),(b,(\delta_1(q'_1,b),\ldots,\delta_1(q'_1,b)))) \in \Delta(((p,\overrightarrow{q}),(a,(q'_1,\ldots,q'_n))))$ pour tout $\overrightarrow{q} \in Q_m$ si et seulement si $push(p',b) \in \Delta(p,a)$. Ainsi, l'information relative aux états contenue dans l'ancien sommet de pile est transférée dans l'état de contrôle et est actualisée dans le nouveau sommet de pile, en simulant l'action de b dans chaque $\mathcal{A}_j$ depuis q'_j.
- $push((p',(\delta_1(i_1,\bot),\ldots,\delta_n(i_n,\bot))),(b,(\delta_1(\delta_1(i_1,\bot),b),\ldots,\delta_n(\delta_n(i_n,\bot),b)))) \in \Delta((p,\overrightarrow{q}),\bot')$ pour tout $\overrightarrow{q} \in Q_m$ si et seulement si $push(p',b) \in \Delta(p,\bot)$. Ainsi, l'information

sur les états des $(\mathcal{A}_i)_{i=1\ldots n}$ est celle relative à la lecture de $\perp$ depuis les états initiaux.

- $skip((p',(q'_1,\ldots,q'_n))) \in \Delta((p,\overrightarrow{q}),(a,(q'_1,\ldots,q'_n)))$ pour tout $\overrightarrow{q} \in Q_m$ si et seulement si $skip(p') \in \Delta(p,a)$.
- $skip((p',(\delta_1(i_1,\perp),\ldots,\delta_n(i_n,\perp)))) \in \Delta((p,\overrightarrow{q}),\perp')$ pour tout $\overrightarrow{q} \in Q_m$ si et seulement si $skip(p') \in \Delta(p,\perp)$.
- $pop((p',(q'_1,\ldots,q'_n))) \in \Delta((p,\overrightarrow{q}),(a,(q'_1,\ldots,q'_n)))$ pour tout $\overrightarrow{q} \in Q_m$ si et seulement si $pop(p') \in \Delta(p,a)$.

On pose $Q'_{\mathbf{E}} = Q_{\mathbf{E}} \times Q_m$ et $Q'_{\mathbf{A}} = Q_{\mathbf{A}} \times Q_m$, et l'on appelle $\mathcal{G}'$ le graphe associé avec $\mathcal{P}'$ et $Q'_{\mathbf{E}} \cup Q'_{\mathbf{A}}$.

D'après le point (4) de la propriété 4.2, il suffit de définir f sur V. Pour un état $p \in Q$ et un mot de pile $u = u_m \cdots u_1 \perp$ sur Γ, on pose $f(p,u) = [(p,\#),v]$, où $v = v_m \cdots v_1$ avec $v_i = (u_i,\overrightarrow{q_i})$ avec $\overrightarrow{q_i} = (q_i^1,\ldots,q_i^n)$, où $i_k,\delta_k(i_k,\perp),q_1^k,\cdots,q_m^k$ est la suite des états de $\mathcal{A}_k$ obtenue en lisant $\widetilde{u} = \perp u_1 \cdots u_m$ depuis i_k. En bref, f se contente de coder, dans la pile, les calculs de chaque automate $\mathcal{A}_k$ sur cette dernière et f n'est rien d'autre qu'une transduction de Γ^* dans Γ'^*.

Enfin, on définit une fonction g de Q' dans $\{1,\ldots,n\} \cup \{\#\}$, et l'on l'étend en un morphisme de Q'^∞ dans $(\{1,\ldots,n\} \cup \{\#\})^\infty$ par :

- $g((p,\#)) = \#$, pour tout $p \in Q$.
- $g((p,(q_1,\ldots,q_n))) = i$, si i est l'unique entier $k = 1,\ldots,n$ tel que $q_k \in F_k$.
- dans tous les autres cas, g n'est pas définie.

et l'on appelle Ω' la condition ω-régulière sur $\mathcal{G}'$ donnée par le langage, ω-régulier (car image par morphisme inverse d'un régulier), $L' = \# \cdot g^{-1}(L)$.

Si on appelle $\mathbb{G}' = (\mathcal{G}',\Omega')$, on vérifie facilement le résultat suivant.

Proposition 4.6 *$\mathbb{G}'$ est un enrichissement déterministe régulier de $\mathcal{G}$.*

De la propriété 4.6 et du corollaire 4.5, on en déduit.

Proposition 4.7 *Dans un jeu muni d'une condition régulière de pile, les ensembles de positions gagnantes sont réguliers.*

4.3.4 Combinaisons booléennes

Dans les propositions 4.1 et 4.2, nous avons vu que les conditions invariantes par translations verticales et par translations horizontales forment une algèbre de Boole. Une question naturelle est de se demander si les combinaisons booléennes de conditions de gain pour lesquelles on peut trouver des enrichissements déterministes réguliers vers des jeux équipés de conditions de gain invariantes par translations, forment une algèbre de Boole. On a alors le résultat suivant.

Théorème 4.4 *Soit $\mathbb{G} = (\mathcal{G},\Omega)$ un jeu sur un graphe de jeu engendré par un processus à pile. Supposons que la condition de gain Ω soit combinaison booléenne de conditions de gain pour lesquelles on peut trouver des enrichissements déterministes réguliers vers des jeux munis de conditions de gain invariantes par translations. Alors on peut trouver un enrichissement déterministe régulier de $\mathbb{G}$ vers un jeu*

équipé d'une condition de gain invariante par translations. Ainsi, les ensembles de positions gagnantes dans $\mathbb{G}$ sont réguliers.

Preuve. La fermeture par complémentation ne pose pas de problème : si Ω est le complémentaire d'une condition $\overline{\Omega}$ pour laquelle on possède un enrichissement déterministe régulier vers un jeu $\mathbb{G}' = (\mathcal{G}',\Omega')$ équipé d'une condition de gain invariante par translations, le jeu $\overline{\mathbb{G}'} = (\mathcal{G}',\overline{\Omega'})$ est un enrichissement déterministe régulier de $\overline{\mathbb{G}} = (\mathcal{G},\overline{\Omega})$, et il est équipé d'une condition de gain invariante par translations.

La fermeture par union, bien que technique si on souhaite la décrire en détail (ce que l'on ne fera pas ici) n'est pas difficile. On appelle $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ le processus à pile sous-jacent à $\mathcal{G}$, et l'on suppose donc que $\Omega = \Omega_1 \cup \Omega_2$ et que l'on possède pour $(\mathcal{G},\Omega_1)$, et $(\mathcal{G},\Omega_2)$ des enrichissements déterministes vers des jeux équipés de conditions invariantes par translation.

Soient f_1 et f_2 les fonctions associées, soient $Q_{m,1}$ et $Q_{m,2}$ les enrichissements des états, soient $\Gamma_{m,1} \ni \bot_{m,1}$ et $\Gamma_{m,2} \ni \bot_{m,2}$ les enrichissements des alphabets, soient Δ_1 et Δ_2 les fonctions de transition et soient Ω'_1 et Ω'_2 les conditions de gains invariantes par translations associées. On considère alors l'enrichissement $\mathcal{P}' = \langle Q',\Gamma',\bot',\Delta'\rangle$ induit par :

- $Q_m = Q_{m,1} \times Q_{m,2}$.
- $\Gamma_m = \Gamma_{m,1} \times \Gamma_{m,2}$.
- $\bot_m = (\bot_{m,1},\bot_{m,2})$.

et où Δ' fonctionne comme Δ_1 sur les composantes $Q_{m,1}$ et $\Gamma_{m,1}$, et comme Δ_2 sur les composantes $Q_{m,2}$ et $\Gamma_{m,2}$.

Enfin, on définit f de telle sorte à ce que f imite f_1 sur les premières composantes, et imite f_2 sur les dernières composantes et l'on pose $\Omega' = \{\Lambda' \mid \tau_1(\Lambda') \in \Omega_1$ ou $\tau_2(\Lambda') \in \Omega_2\}$, où τ_1 (respectivement τ_2) désigne la projection où l'on oublie les composantes relatives à Q_2 et Γ_2 (respectivement à Q_1 et Γ_1).

On vérifie alors que l'on obtient bien de cette façon un enrichissement déterministe régulier vers un jeu muni d'une condition de gain invariante par translations.

La fermeture par intersection se fait de la même façon, sauf que dans ce cas on pose $\Omega' = \{\Lambda' \mid \tau_1(\Lambda') \in \Omega_1$ et $\tau_2(\Lambda') \in \Omega_2\}$. □

Chapitre 5

Conditions de parité et d'explosion

Sommaire

Dans ce chapitre, on explique comment calculer les ensembles de retours dans des jeux sur des graphes de processus à pile munis de diverses conditions de gain.

Dans un premier temps, on étudie les conditions de parité, d'explosion stricte, et de parité en escalier. Pour cela, on introduit la notion de M/B factorisation, qui permet de caractériser les parties gagnantes. Les nombreux points communs entre ces conditions de gain permettent de donner une méthode générique pour calculer les ensembles de retours, ce qui permet, grâce aux résultats du chapitre 4, de calculer l'ensemble des positions gagnantes. On se concentre particulièrement sur la condition de parité, et l'on décrit la construction, à partir d'un jeu sur un graphe de processus

à pile, d'un jeu de parité sur un graphe fini de taille exponentielle dans lequel les positions gagnantes pour Eve permettent de déduire les ensembles de retours dans le premier jeu. Si cette construction est proche de celle donnée par I. Walukiewicz dans [83, 84], la preuve est en revanche assez différente.

De la construction pour la condition de parité, on déduit une construction similaire pour la condition d'explosion stricte. On s'intéresse ensuite à la condition d'explosion, et l'on montre qu'Eve possède une stratégie positionnelle gagnante dans ce cas, ce qui permet de montrer que les positions gagnantes pour une condition d'explosion sont les mêmes que pour une condition d'explosion stricte.

Pour ce qui est de la condition de parité en escalier, on adapte la construction donnée pour la condition de parité. Cela permet en particulier de construire des stratégies gagnantes à pile. Comme de telles stratégies utilisent une mémoire infinie, on donne un exemple de jeu muni d'une condition de parité en escalier dans lequel les stratégies gagnantes pour Eve nécessitent une mémoire infinie. Ce dernier résultat montre en quelque sorte l'optimalité des stratégies à pile.

Toutes les constructions précédentes conduisent à des algorithmes fonctionnant en temps exponentiel. On montre que ces algorithmes sont optimaux, à savoir que décider le gagnant dans de tels jeux est DEXPTIME-complet. Pour cela, on rappelle la preuve donnée par I. Walukiewicz dans [83, 84] pour les conditions de parité. Il est alors facile de voir qu'elle s'adapte aux autres conditions de gain.

Dans tout ce qui suit, on considère un processus à pile sans alphabet d'entrée, que l'on note $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$. On appelle $G = (V,E)$ le graphe qu'il engendre. On se donne une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de Q et l'on considère le graphe de jeu $\mathcal{G} = (G,V_{\mathbf{E}},V_{\mathbf{A}})$ associé à G et à cette partition de Q. Enfin, on se donne une condition de gain Ω et l'on appelle $\mathbb{G}$ le jeu $(\mathcal{G},\Omega)$.

5.1 Factorisation des parties

5.1.1 M/B factorisation

Supposons que Ω soit de l'une des formes suivantes : parité, explosion stricte ou parité en escalier.

Considérons une partie Λ dans $\mathbb{G}$ et sa représentation graphique, comme illustré dans la figure 5.1.

Quelle est la forme des parties gagnées par Eve dans un jeu muni de la condition d'explosion stricte? Il s'agit des parties dans lesquelles toute hauteur de pile n'apparaît qu'un nombre fini de fois, c'est-à-dire que tout niveau de pile est un jour quitté pour toujours. Etant donnée une partie (partielle ou non) $\Lambda = v_0v_1v_2\cdots$ dans $\mathbb{G}$, et un entier $h \geq 1$, un facteur $v_i \cdots v_j$ de Λ est une *bosse sur le niveau h* si et seulement si $|v_i| = |v_j| = h$ et pour tout $i < k < j$, $|v_k| > h$. Par opposition aux bosses, on appelle *marche quittant le niveau h* un facteur de Λ, v_iv_{i+1}, de longueur 2, tel que $|v_i| = h$, $|v_{i+1}| = h+1$, et pour tout $i+1 \leq k < |\Lambda|$, $|v_k| > h$. En d'autres termes, une marche est un facteur de longueur 2 qui n'est pas préfixe d'une bosse.

Il est alors naturel, étant donnée une partie Λ dans $\mathbb{G}$, de considérer les positions suivantes.

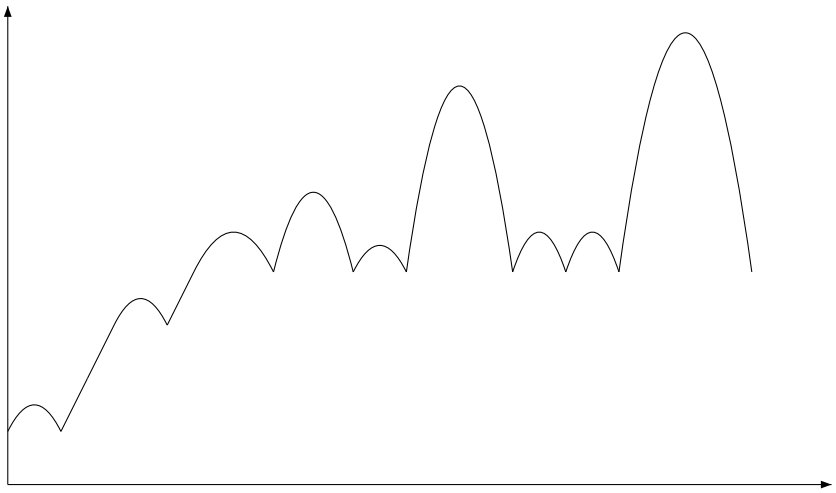


FIG. 5.1 – *Représentation graphique d'une partie*

Définition 5.1 *Etant donnée une partie (partielle ou non)* $\Lambda = v_0 v_1 v_2 \cdots$ *dans* $\mathbb{G}$*, on note* $Steps_\Lambda$ *le sous-ensemble d'indices correspondant à des configurations de hauteur de pile minimale par rapport au futur de la partie. Plus précisément :*

$$Steps_\Lambda = \{n \in \mathbb{N} \mid \forall m \geq n,\ |v_m| \geq |v_n|\},$$

où $|v_n|$ *désigne la hauteur de la pile dans la configuration* v_n*.*

Ainsi, l'ensemble $Steps_\Lambda$ induit une factorisation naturelle de Λ en bosses et en marches.

Définition 5.2 (M/B factorisation) *Etant donnée une partie (partielle ou non)* $\Lambda = v_0 v_1 v_2 \cdots$*, on appelle* Marche/Bosse factorisation *de* Λ *(ou plus simplement* M/B factorisation*) la suite (finie or infinie)* $(\Lambda_i)_{i \geq 0}$ *de facteurs de* Λ *définis de la façon suivante. Soit* $Steps_\Lambda = \{n_0 < n_1 < n_2 < \cdots\}$*, on pose alors, pour tout* $0 \leq i < |Steps_\Lambda|$*,* $\Lambda_i = v_{n_i} \cdots v_{n_{i+1}}$*. Ainsi, pour tout* $i \geq 0$*, le premier sommet de* Λ_{i+1} *est le dernier sommet de* Λ_i*. De plus* $\Lambda = \Lambda_1 \odot \Lambda_2 \odot \Lambda_3 \odot \cdots$ *où* $\Lambda_i \odot \Lambda_{i+1}$ *désigne la concaténation de* Λ_i *et de* Λ_{i+1} *privée de son premier sommet.*

Enfin, on qualifiera un facteur Λ_i *de* marche *si* $|v_{n_{i+1}}| = |v_{n_i}| + 1$ *et sinon de* bosse *(et dans ce cas,* $|v_{n_{i+1}}| = |v_{n_i}|$*).*

La figure 5.2 illustre la M/B factorisation de la partie représentée dans la figure 5.1.

5.1.2 De l'utilité de la M/B factorisation

Etant donnée une partie Λ dans un jeu muni d'une condition de parité, on appelle *couleur* d'un facteur de la M/B factorisation de Λ la plus petite couleur apparaissant dans le facteur.

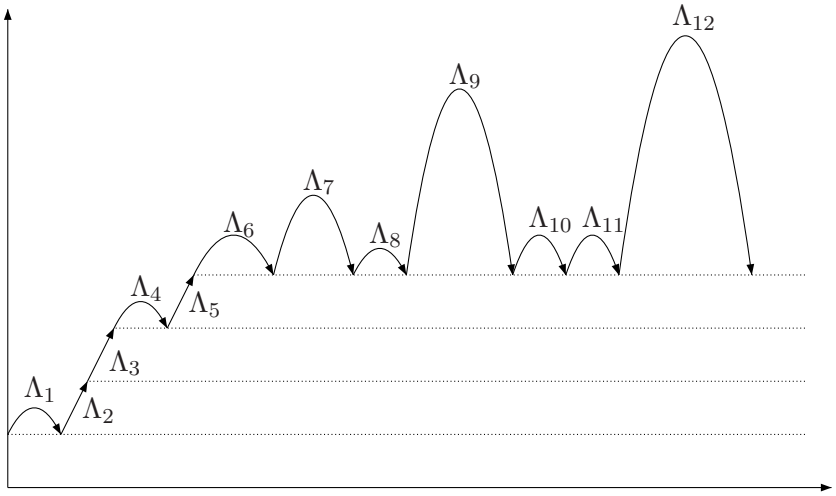


FIG. 5.2 – *M/B factorisation*

La M/B factorisation permet de décider si une partie est remportée par Eve pour les conditions de gains étudiées dans ce chapitre.

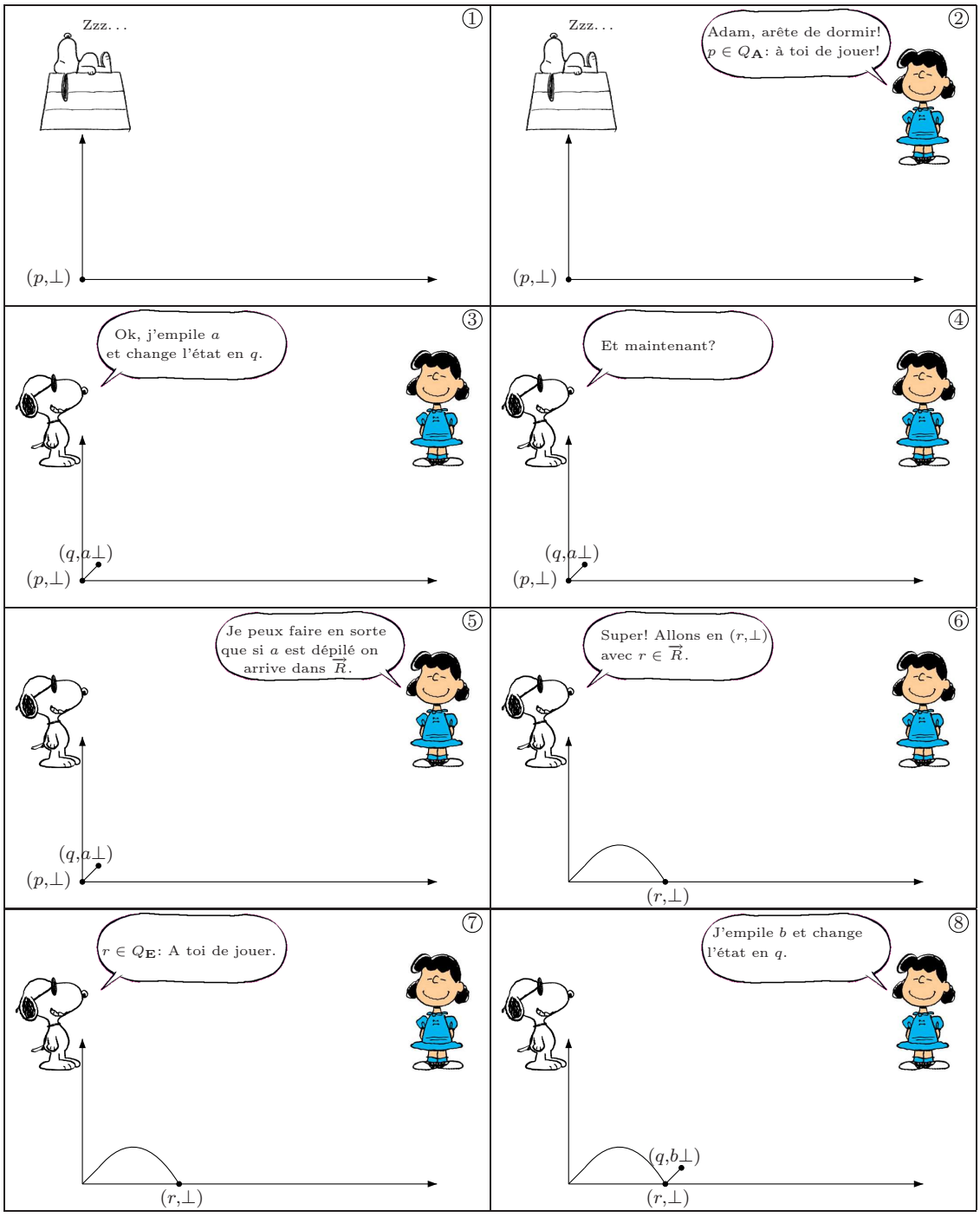
Proposition 5.1 *Une partie Λ dans $\mathbb{G}$ est une partie gagnée par Eve si et seulement si Λ est infinie, et selon les cas :*

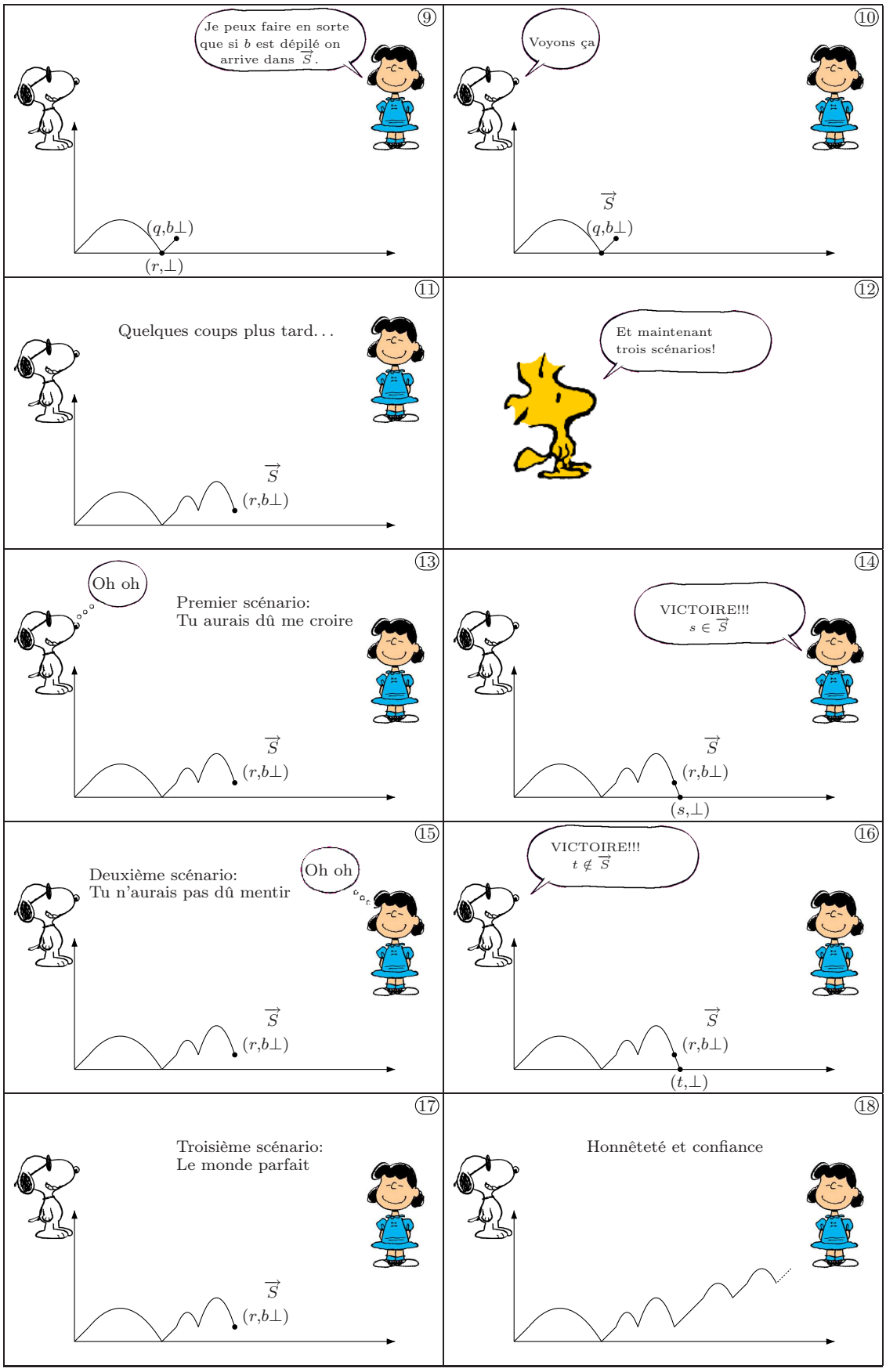
- *si Ω est une condition d'explosion stricte : il existe une infinité de marches dans la M/B factorisation de Λ.*
- *si Ω est une condition de parité : la plus petite couleur apparaissant infiniment souvent dans la suite des couleurs des facteurs de la M/B factorisation de Λ est paire.*
- *si Ω est une condition de parité en escalier : la plus petite couleur apparaissant infiniment souvent dans la suite des couleurs des premiers sommets des facteurs de la M/B factorisation de Λ est paire.*

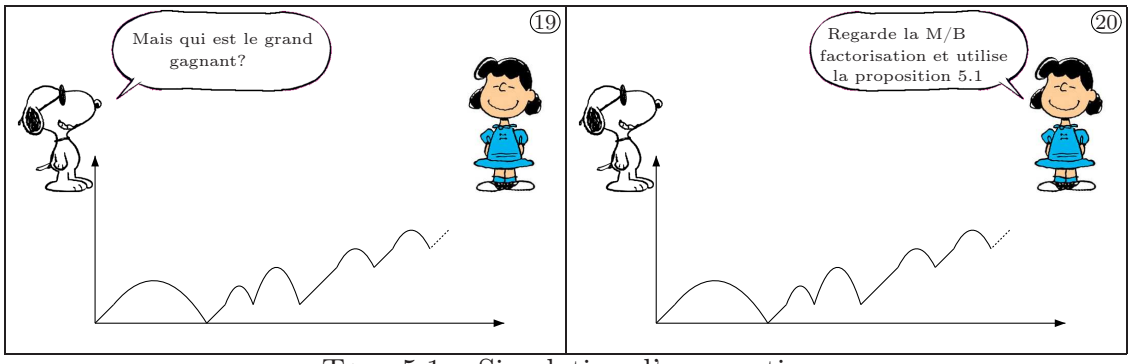
Les conditions d'explosion stricte, de parité, et de parité en escalier étant invariantes par translations, pour calculer l'ensemble des positions gagnantes, il suffit de calculer les ensembles de retours, c'est-à-dire de décider pour tout $p \in Q$, tout $a \in \Gamma$, et tout $R \subseteq Q$, si $(p,a\bot)$ est gagnante pour Eve dans le jeu $\mathbb{G}(R)$. De plus, on doit également décider, pour tout $p \in Q$, si la configuration $(p,\bot)$ est gagnante pour Eve dans $\mathbb{G}$. D'après la proposition 5.1, il suffit de connaître la nature de la M/B factorisation d'une partie pour savoir par qui cette dernière est remportée, et donc la question de savoir si Eve possède une stratégie gagnante revient à savoir si Eve peut jouer de telle façon que la M/B factorisation de la partie obtenue vérifie les conditions établies dans la proposition 5.1. Nous expliquons comment décider si une configuration de la forme $(p,\bot)$ est gagnante pour Eve dans $\mathbb{G}$. Le cas des configurations de la forme $(p,a\bot)$ dans les jeux $\mathbb{G}(R)$ découle du cas précédent.

De façon assez informelle, imaginons maintenant un nouveau jeu dans lequel Eve et Adam se mettent directement d'accord sur la M/B factorisation et ses propriétés.

Dans un premier temps, on suppose que la condition de gain considérée est, soit une condition d'explosion stricte, soit une condition de parité en escalier. Cette simulation est illustrée par la bande dessinée donnée dans la table 5.1.




⑨
Je peux faire en sorte que si b est dépilé on arrive dans $\overrightarrow{S}$.
$(q,b\perp)$
$(r,\perp)$
⑩
Voyons ça
$\overrightarrow{S}$
$(q,b\perp)$
⑪
Quelques coups plus tard...
$\overrightarrow{S}$
$(r,b\perp)$
⑫
Et maintenant trois scénarios!
⑬
Oh oh
Premier scénario:
Tu aurais dû me croire
$\overrightarrow{S}$
$(r,b\perp)$
⑭
VICTOIRE!!!
$s \in \overrightarrow{S}$
$\overrightarrow{S}$
$(r,b\perp)$
$(s,\perp)$
⑮
Deuxième scénario:
Tu n'aurais pas dû mentir
Oh oh
$\overrightarrow{S}$
$(r,b\perp)$
⑯
VICTOIRE!!!
$t \notin \overrightarrow{S}$
$\overrightarrow{S}$
$(r,b\perp)$
$(t,\perp)$
⑰
Troisième scénario:
Le monde parfait
$\overrightarrow{S}$
$(r,b\perp)$
⑱
Honnêteté et confiance



TAB. 5.1 – Simulation d'une partie

Afin de simuler une partie dans le jeu $\mathbb{G}$ depuis une configuration $(p,\bot)$, Eve et Adam procèdent de la façon suivante. Le joueur à qui appartient la configuration $(p,\bot)$ peut choisir, a priori, d'empiler, de dépiler, ou de ne pas modifier la pile (vignettes 1 à 2).

Dans le cas où il ne modifie pas la pile, c'est-à-dire qu'une règle de la forme $skip(q) \in \Delta(p,\bot)$ est appliquée, le facteur correspondant est la bosse triviale $(p,\bot)(q,\bot)$.

Dans le cas où il décide d'empiler (vignette 3), c'est-à-dire d'appliquer une règle de la forme $push(q,a) \in \Delta(p,\bot)$, on souhaite décider si $(p,\bot)(q,a\bot)$ sera une marche ou le préfixe d'une bosse. Bien sûr, cela dépend de la manière dont Eve et Adam vont décider de jouer. Eve donne alors à Adam des informations sur la stratégie qu'elle va adopter, à savoir un sous-ensemble $R \subseteq Q$ d'états r pouvant être atteints si le a est dépilé (vignette 5). Adam peut alors prolonger la partie en une bosse (vignette 6), et donc continuer la partie depuis une position, de son choix, $(r,\bot)$ avec $r \in R$ (vignette 7 à 9), ou continuer avec la nouvelle lettre comme sommet de pile (vignettes 10 avec b comme sommet de pile et S comme ensemble). Dans ce dernier cas, la partie se prolonge depuis $(q,b\bot)$ si b est le nouveau sommet de pile. Si b est un jour dépilé, Eve remporte la partie (vignettes 13 et 14) si l'état de contrôle est dans S (S est l'ensemble annoncé par Eve précédemment), et sinon c'est Adam qui gagne (vignettes 15 et 16). Ainsi, la nature de ce second choix est double du point de vue d'Adam : il peut lui permettre de prouver que l'ensemble S donné par Eve est trop petit (il existe un état qui n'est pas dans S et qui est obtenu en dépilant b), ou simplement de simuler une partie où b ne sera jamais dépilé.

Quel que soit le choix d'Adam, les deux joueurs se sont mis d'accord sur le premier facteur de la M/B factorisation. De plus, il est facile pour eux de décider s'il s'agit d'une marche ou d'une bosse, et dans le cas où Ω est une condition de parité en escalier, de savoir la couleur du dernier sommet. Dès lors, au cours d'une partie infinie (vignettes 17 à 20), ils peuvent décider qui est le gagnant, puisque la condition sur la M/B factorisation est une condition de Büchi si Ω est une condition d'explosion stricte (il faut qu'il y ait une infinité de marche dans la M/B factorisation), et c'est une condition de parité si Ω est une condition de parité en escalier.

Plus formellement, la construction précédente n'est que la transcription du fait

suivant (qui est à rapprocher du lemme 4.3 du chapitre 4).

Proposition 5.2 *Considérons un ensemble $R \subseteq Q$, un état de contrôle $p \in Q$, et une lettre $a \in \Gamma$. La configuration $(p,a\perp)$ est gagnante pour Eve dans le jeu $\mathbb{G}(R)$ si et seulement si :*

- **si** $p \in Q_{\mathbf{E}}$ **:** *soit il existe une règle $pop(r) \in \Delta(p,a)$ avec $r \in R$, soit il existe une règle $skip(q) \in \Delta(p,a)$, et $(q,a\perp)$ est gagnante pour Eve dans $\mathbb{G}(R)$, soit il existe une règle $push(q,b) \in \Delta(p,a)$, et un ensemble $S \subseteq Q$ tel que $(q,ba\perp)$ est une position gagnante pour Eve dans le jeu $\mathbb{G}(S)$, et pour tout $s \in S$, $(s,a\perp)$ est une position gagnante pour Eve dans le jeu $\mathbb{G}(R)$.*
- **si** $p \in Q_{\mathbf{A}}$ **:** *pour toute règle $pop(r) \in \Delta(p,a)$, $r \in R$, pour toute règle $skip(q) \in \Delta(p,a)$, $(q,a\perp)$ est une position gagnante pour Eve dans $\mathbb{G}(R)$, et pour toute règle $push(q,b) \in \Delta(p,a)$, il existe un ensemble $S \subseteq Q$ tel que $(q,ba\perp)$ est une position gagnante pour Eve dans le jeu $\mathbb{G}(S)$, et pour tout $s \in S$, $(s,a\perp)$ est une position gagnante pour Eve dans le jeu $\mathbb{G}(R)$.*

Evoquons maintenant le cas de la condition de parité, qui est un peu plus compliqué que le précédent puisque l'on doit connaître la couleur de chaque facteur de la M/B factorisation afin de déterminer le gagnant. On considère un jeu comme précédemment, sauf que cette fois Eve raffine l'information qu'elle donne à Adam. Elle ne se contente pas de donner un ensemble R d'états atteignables lors d'un dépilement de a, mais donne un vecteur $\overrightarrow{R} = (R_0, \dots R_d)$, où $\{0, \dots d\}$ sont les couleurs intervenant dans la condition de parité, tel que R_i désigne l'ensemble des états atteignables lors d'un dépilement de a, si la plus petite couleur visitée lorsque a est dans la pile est i. Lorsqu'Adam choisit de faire une bosse, la couleur i de celle-ci est déterminée par le choix d'Adam de continuer la partie depuis une configuration d'état de contrôle r avec $r \in S_i$ (la couleur est alors i). Dans le cas d'un dépilement, le vainqueur est déterminé selon l'appartenance de l'état de contrôle obtenu en dépilant à la composante d'indice θ du vecteur d'ensembles donné par Eve, où θ désigne la plus petite couleur visitée sur le niveau quitté en dépilant.

Nous ne donnons pas ici de preuve du fait que les jeux ainsi décrits sont équivalents à $\mathbb{G}$ (et à ses variantes $\mathbb{G}(R)$), puisque ces jeux sont décrits pour donner l'intuition de la construction qui va suivre. Cependant, les résultats qui vont suivre peuvent être facilement adaptés afin de prouver ces équivalences.

5.2 Réduction : la condition de parité

Dans les jeux précédemment décrits, on peut remarquer qu'à tout moment la seule information utile est le sommet de pile, l'état de contrôle de la configuration courante, ainsi que le vecteur de sous-ensembles de Q fourni par Eve. Cette information étant finie, l'idée est de coder les jeux précédents dans un jeu sur un graphe fini.

On pourrait proposer une construction et une preuve commune pour les conditions de parité, d'explosion stricte, et de parité en escalier. Cependant, bien que ces conditions de gains soient liées, leur nature est différente, et une construction commune masquerait le côté intuitif des constructions proposées. C'est pourquoi

nous allons dans un premier temps proposer une solution pour les conditions de parité. Ensuite, nous verrons comment l'adapter (et la simplifier) pour obtenir la construction dans le cas des conditions d'explosion stricte et des conditions de parité en escalier.

On suppose donc qu'il existe une fonction de coloriage ρ de Q dans un ensemble de couleurs $\{0,\ldots,d\}$, et que Ω est la condition de parité associée.

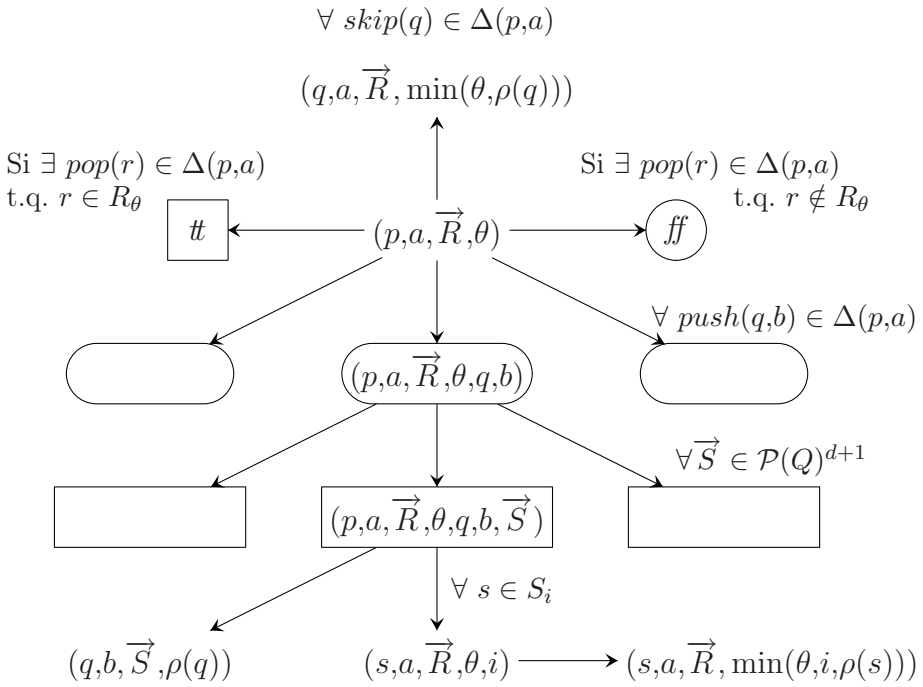


FIG. 5.3 – *Structure locale de $\widetilde{\mathcal{G}}$ pour la condition de parité.*

On considère le graphe de jeu $\widetilde{\mathcal{G}}$ illustré dans la figure 5.3, et dont voici une description :

- les sommets principaux de $\widetilde{\mathcal{G}}$ sont ceux de la forme $(p,a,\overrightarrow{R},\theta)$, où $p \in Q$, $a \in \Gamma$, $\overrightarrow{R} = (R_0,\ldots,R_d) \in \mathcal{P}(Q)^{d+1}$ et $\theta \in \{0,\ldots,d\}$. Un sommet $(p,a,\overrightarrow{R},\theta)$ représente une partie partielle Λ dans $\mathbb{G}$ telle que :
  - le dernier sommet de Λ est de la forme (p,au) pour un certain $u \in \Gamma^*$.
  - Eve prétend pouvoir jouer depuis Λ de telle sorte que si a est un jour dépilé, l'état de contrôle atteint après avoir dépilé a est dans R_m, où m est la plus petite couleur vue lorsque a se trouvait dans la pile.
  - la plus petite couleur vue depuis le moment où a a été empilé est θ.

  Un sommet de la forme $(p,a,\overrightarrow{R},\theta)$ appartient à Eve si et seulement si $p \in Q_{\mathbf{E}}$.
- les états *tt* et *ff* sont là pour s'assurer que les vecteurs $\overrightarrow{R}$ dans les sommets principaux sont corrects. Le sommet *tt* appartient à Adam, tandis que *ff* appartient à Eve. Ces sommets étant des culs-de-sac, une partie qui arrive dans

tt sera remportée par Eve tandis qu'une partie arrivant dans *ff* sera remportée par Adam.

Il y a une transition d'un sommet $(p,a,\overrightarrow{R},\theta)$ vers *tt*, si et seulement s'il existe une règle de la forme $pop(r) \in \Delta(p,a)$, telle que $r \in R_\theta$ (ce qui traduit le fait que $\overrightarrow{R}$ est correct par rapport à cette règle). Symétriquement, il y a une transition d'un sommet $(p,a,\overrightarrow{R},\theta)$ vers *ff* si et seulement s'il existe une règle de la forme $pop(r) \in \Delta(p,a)$, telle que $r \notin R_\theta$ (ce qui traduit le fait que $\overrightarrow{R}$ n'est pas correct par rapport à cette règle).

– afin de simuler l'application d'une transition $skip(q) \in \Delta(p,a)$, le joueur à qui appartient $(p,a,\overrightarrow{R},\theta)$ va en $(q,a,\overrightarrow{R},\min(\theta,\rho(q)))$. Remarquons que dans ce cas la dernière composante doit être actualisée, puisque la plus petite couleur vue depuis que a est dans la pile est désormais $\min(\theta,\rho(q))$.
– afin de simuler l'application d'une transition $push(q,b) \in \Delta(p,a)$, le joueur à qui appartient $(p,a,\overrightarrow{R},\theta)$ va en $(p,a,\overrightarrow{R},\theta,q,b)$. Ce dernier sommet appartient à Eve et celle-ci doit donc choisir un vecteur $\overrightarrow{S} = (S_0,\ldots,S_d) \in \mathcal{P}(Q)^{d+1}$ décrivant les états pouvant être atteints si b est un jour dépilé. Eve va alors dans le sommet $(p,a,\overrightarrow{R},\theta,q,b,\overrightarrow{S})$.

  Ce dernier appartient à Adam. Depuis $(p,a,\overrightarrow{R},\theta,q,b,\overrightarrow{S})$, Adam choisit s'il veut simuler une bosse ou une marche. Dans le premier cas, il va dans un sommet $(s,a,\overrightarrow{R},\theta,i)$, pour un certain $i \in \{0,\ldots,d\}$ et un $s \in S_i$, et de là dans $(s,a,\overrightarrow{R},\min(\theta,i,\rho(s)))$. Dans le second cas, Adam va dans le sommet $(q,b,\overrightarrow{S},\rho(q))$. Dans le premier cas la dernière composante est mise à jour, on vient de faire une bosse de couleur i et donc la plus petite couleur visitée depuis que a a été empilé est désormais $\min(\theta,i,\rho(s))$. Dans le second cas on initialise la dernière composante, la seule couleur vue depuis que b a été empilé étant $\rho(q)$.

Le graphe de jeu $\widetilde{\mathcal{G}}$ est muni d'un coloriage (partiel) sur les sommets : les sommets colorés sont ceux de la forme $(p,a,\overrightarrow{R},\theta)$ et ceux de la forme $(p,a,\overrightarrow{R},\theta,i)$, avec $p \in Q$, $a \in \Gamma$, $\overrightarrow{R} \in \mathcal{P}(Q)^{d+1}$ et $0 \leq \theta,i \leq d$. Les premiers ont la couleur $\rho(p)$ et les seconds la couleur i (puisqu'ils codent des bosses de couleurs i).

Remarque 5.1 On peut objecter que le jeu $\widetilde{\mathbb{G}}$ n'est pas exactement conforme à la définition classique d'un jeu de parité, puisque $\widetilde{\mathcal{G}}$ n'est que partiellement coloré. Cependant, en remarquant que tout chemin infini dans $\widetilde{\mathcal{G}}$ visite une infinité de sommets colorés, on se ramène facilement à un jeu équivalent en introduisant une couleur maximale supplémentaire dont on équipe tous les sommets qui n'étaient pas colorés dans le graphe précédent. Une telle couleur n'interfère pas dans la décision finale, puisqu'elle ne peut être la plus petite couleur infiniment souvent visitée.

Si l'on appelle $\widetilde{\mathbb{G}}$ le jeu de parité sur $\widetilde{\mathcal{G}}$, on a le résultat suivant.

Théorème 5.1 *Soit $p_{in} \in Q$, soit $a_{in} \in \Gamma$ et soit $T \subseteq Q$. On a les équivalences suivantes :*

– *Eve possède une stratégie gagnante dans $\mathbb{G}(T)$ depuis $(p_{in},a_{in}\bot)$ si et seulement*

si elle possède une stratégie gagnante dans $\widetilde{\mathbb{G}}$ *depuis* $(p_{in},a_{in},(T,\dots,T),\rho(p_{in}))$.

– *Eve possède une stratégie gagnante dans* $\mathbb{G}$ *depuis* $(p_{in},\bot)$ *si et seulement si elle possède une stratégie gagnante dans* $\widetilde{\mathbb{G}}$ *depuis* $(p_{in},\bot,(\emptyset,\dots,\emptyset),\rho(p_{in}))$.

De plus, on peut construire une stratégie gagnante à pile dans $\mathbb{G}$ *pour Eve depuis toute position gagnante.*

Les preuves de ces deux points étant similaires, nous nous contentons de celle du premier. L'existence de stratégies à pile est une conséquence de la preuve et des résultats du paragraphe 4.2.5 sur la composition des stratégies.

5.2.1 Factorisation dans $\widetilde{\mathbb{G}}$

Remarquons tout d'abord qu'une partie dans $\widetilde{\mathbb{G}}$ visite *régulièrement*, au plus tous les quatre coups, des sommets de la forme $(p,a,\overrightarrow{R},\theta)$. On qualifie de *round* la séquence de sommets entre deux passages dans de telles positions. Un round peut alors être de plusieurs types :

– il est de la forme $(p,a,\overrightarrow{R},\theta)(p',a,\overrightarrow{R},\theta)$ et correspond donc à la simulation d'une règle de type skip. On parle alors de *bosse (triviale)*.

– il est de la forme $(p,a,\overrightarrow{R},\theta)(p,a,\overrightarrow{R},\theta,q,b)(p,a,\overrightarrow{R},\theta,q,b,\overrightarrow{S})(s,a,\overrightarrow{R},\theta,i)$ $(s,a,\overrightarrow{R},\min(\theta,i,\rho(s)))$ et correspond donc à la simulation d'une règle empilant un b suivie d'une série de coups se terminant par le dépilement du b. On le qualifie de *bosse*.

– il est de la forme $(p,a,\overrightarrow{R},\theta)(p,a,\overrightarrow{R},\theta,q,b)(p,a,\overrightarrow{R},\theta,q,b,\overrightarrow{S})(q,b,\overrightarrow{S},\rho(q))$ et correspond donc à la simulation d'une règle d'empilement d'un b qui ne sera pas dépilé. On le qualifie de *marche*.

Etant donnée une partie $\lambda = v_0v_1v_2\cdots$ dans $\widetilde{\mathbb{G}}$, on peut considérer le sous-ensemble d'indices correspondant à des sommets de la forme $(p,a,\overrightarrow{R},\theta)$. Plus précisément :

$$Steps_\lambda = \{n \in \mathbb{N} \mid v_n = (p,a,\overrightarrow{R},\theta),\ p \in Q,\ a \in \Gamma,\ \overrightarrow{R} \in \mathcal{P}(Q)^{d+1},\ 0 \leq \theta \leq d\}$$

Tout comme dans le cas des jeux sur un graphe de processus à pile, l'ensemble $Steps_\lambda$ induit une factorisation naturelle d'une partie λ en bosses et en marches.

Définition 5.3 (M/B factorisation) *Etant donnée une partie, partielle ou non,* $\lambda = v_0v_1v_2\cdots$ *dans* $\widetilde{\mathbb{G}}$*, on appelle* Marche/Bosse factorisation *de* λ*, ou plus simplement* M/B factorisation*, la suite, finie ou infinie,* $(\lambda_i)_{i\geq 0}$ *de rounds de* λ *définis de la façon suivante. Soit* $Steps_\lambda = \{n_0 < n_1 < n_2 < \cdots\}$*, on pose alors, pour tout* $0 \leq i < |Steps_\lambda|$, $\lambda_i = v_{n_i}\cdots v_{n_{i+1}}$.

Ainsi, pour tout $i \geq 0$*, le premier sommet de* λ_{i+1} *est le dernier sommet de* λ_i*. De plus,* $\lambda = \lambda_1 \odot \lambda_2 \odot \lambda_3 \odot \cdots$*, où* $\lambda_i \odot \lambda_{i+1}$ *désigne la concaténation de* λ_i *et de* λ_{i+1} *privé de son premier sommet.*

Enfin, la couleur *d'un facteur désigne la plus petite couleur des sommets qui le composent.*

Afin de prouver les deux implications du théorème 5.1, on construira à partir d'une stratégie gagnante pour Eve dans l'un des deux jeux, une stratégie dans le second jeu. Cette stratégie utilise une mémoire qui code une partie dans le premier jeu, où Eve respecte sa stratégie gagnante, et qui est donc une partie gagnante. L'argument principal pour prouver que la nouvelle stratégie est gagnante consiste essentiellement à montrer une correspondance entre les M/B factorisations des deux parties, et à conclure en utilisant le fait que la première partie est gagnante.

5.2.2 Implication directe

Supposons que la configuration $(p_{in},a_{in}\bot)$ soit gagnante dans le jeu $\mathbb{G}(T)$ et considérons une stratégie Φ gagnante pour Eve dans $\mathbb{G}(T)$ depuis $(p_{in},a_{in}\bot)$.

Remarque 5.2 Nous pourrions prendre pour Φ une stratégie positionnelle, puisque $\widetilde{\mathbb{G}}$ est un jeu de parité. Cependant, la preuve ne serait pas foncièrement simplifiée et surtout cette dernière ne pourrait pas être adaptée pour la condition de parité en escalier pour laquelle il n'existe pas toujours de stratégie sans mémoire.

A partir de Φ, on définit une stratégie φ pour Eve dans $\widetilde{\mathbb{G}}$ depuis $(p_{in},a_{in},(T,\dots,T),\rho(p_{in}))$. La stratégie utilise une mémoire qui contient une partie partielle dans $\mathbb{G}$, c'est-à-dire un élément de V^*, et son contenu sera noté Λ. Au départ, Λ est réduit au sommet $(p_{in},a_{in}\bot)$. On commence par décrire φ, puis on explique comment Λ est mise à jour. La stratégie φ, ainsi que la mise à jour de Λ sont décrites pour un round.

Choix du coup : On suppose donc que l'on est dans une configuration de la forme $(p,a,\overrightarrow{R},\theta)$ avec $p \in Q_{\mathbf{E}}$. Le coup donné par φ dépend alors de $\Phi(\Lambda)$:

- si $\Phi(\Lambda) = pop(r)$, alors Eve va dans $\#$ (la proposition 5.3 montrera que ce coup est toujours possible).
- si $\Phi(\Lambda) = skip(q)$, alors Eve va dans $(q,a,\overrightarrow{R},\min(\theta,\rho(q)))$.
- si $\Phi(\Lambda) = push(q,b)$, alors Eve va dans $(p,a,\overrightarrow{R},\theta,q,b)$.

Dans ce dernier cas, ou lorsque $p \in Q_{\mathbf{A}}$ et qu'Adam est allé en $(p,a,\overrightarrow{R},\theta,q,b)$, nous devons également expliquer comment Eve joue depuis $(p,a,\overrightarrow{R},\theta,q,b)$. Elle doit choisir un vecteur $\overrightarrow{S} \in \mathcal{P}(Q)^{d+1}$ qui décrit les états atteignables si b est dépilé en fonction de la plus petite couleur visitée entre temps. Afin de définir $\overrightarrow{S}$, Eve considère l'ensemble des parties possibles dans $\mathbb{G}$, où elle respecte sa stratégie Φ, et qui commencent par $\Lambda \cdot (q,bau)$, si (p,au) désigne le dernier sommet de Λ. Pour chacune de ces parties, Eve regarde si une configuration de la forme (s,au) apparaît après $\Lambda \cdot (q,bau)$, c'est-à-dire si b est un jour dépilé. Si tel est le cas, Eve considère la première configuration (s,au) apparaissant après $\Lambda \cdot (q,bau)$ ainsi que la plus petite couleur visitée i lorsque b se trouvait dans la pile. Pour tout $i \in \{0,\dots d\}$, S_i est exactement l'ensemble des $s \in Q$, tel que l'on ait la propriété précédente. Enfin, on pose $\overrightarrow{S} = (S_0,\dots,S_d)$ et Eve joue alors vers $(p,a,\overrightarrow{R},\theta,q,b,\overrightarrow{S})$.

Mise à jour de Λ : La mise à jour de Λ est effectuée chaque fois que l'on arrive dans un sommet de la forme $(p,a,\overrightarrow{R},\theta)$. On a alors trois cas, selon la nature

du round :

- le round est une bosse triviale et l'on vient donc de simuler une action $skip(q)$. Si l'on appelle (p,au) le dernier sommet de Λ, on augmente Λ du sommet (q,au).
- le round est une bosse et l'on vient donc de simuler une bosse commençant par une action $push(q,b)$ et se terminant dans un état s. Si l'on appelle (p,au) le dernier sommet de Λ, on met à jour Λ en y ajoutant (q,bau) suivi d'une série de coups, où Eve respecte Φ, et qui conduisent à dépiler b et à arriver dans (s,au), tout en visitant i comme plus petite couleur lorsque b est dans la pile, où i est la couleur de l'avant-dernier sommet du round (c'est-à-dire que $s \in S_i$, si $\overrightarrow{S}$ désigne le vecteur donné par Eve lors du round).
- le round est une marche et l'on vient donc de simuler une action $push(q,b)$. Si l'on appelle (p,au) le dernier sommet de Λ, on augmente Λ du sommet (q,bau).

Dès lors, à toute partie partielle λ dans $\widetilde{\mathbb{G}}$ dans laquelle Eve respecte sa stratégie φ, est associée une partie partielle Λ dans $\mathbb{G}$. Une récurrence immédiate permet de prouver que Λ est une partie dans laquelle Eve respecte Φ. Le même raisonnement peut être mené lorsque λ est une partie infinie, et la partie Λ associée est alors infinie, commence en $(p_{in},a_{in}\bot)$ et lors de celle-ci, Eve respecte sa stratégie gagnante Φ. En particulier, la plus petite couleur infiniment souvent visitée dans Λ est donc paire.

La proposition suivante découle directement de la définition de φ

Proposition 5.3 *Soit λ une partie partielle dans $\widetilde{\mathbb{G}}$ commençant en $(p_{in},a_{in},(T,\ldots,T),\rho(p_{in}))$, se terminant dans un sommet de la forme $(p,a,\overrightarrow{R},\theta)$, et où Eve respecte φ. Soit Λ la partie associée à λ construite par la stratégie φ. On a alors les points suivants :*

1. *Λ se termine dans un sommet de la forme (p,au) pour un certain $u \in \Gamma^*$.*
2. *θ est la plus petite couleur visitée dans Λ, depuis que a a été empilé.*
3. *si la partie Λ est prolongée en une partie où Eve respecte Φ, alors, si a est dépilé, la configuration (r,u) alors atteinte est telle que $r \in R_i$, où i est la plus petite couleur visitée lorsque a était dans la pile.*

Remarque 5.3 La proposition 5.3 implique que la stratégie φ est bien définie lorsqu'elle donne un coup vers tt. De même, on en déduit que si Eve respecte φ, le sommet ff ne peut être atteint.

La remarque précédente règle donc le cas des parties finies où Eve respecte φ : elle aboutissent dans le sommet tt et sont alors remportées par Eve.

Des définitions de $\widetilde{\mathcal{G}}$ et de φ, on déduit la proposition suivante.

Proposition 5.4 *Soit λ une partie infinie dans $\widetilde{\mathbb{G}}$ commençant en $(p_{in},a_{in},(T,\ldots,T),\rho(p_{in}))$ où Eve respecte φ. Soit Λ la partie associée à λ construite par la stratégie φ. Soient $(\lambda_i)_{i\geq 1}$ et $(\Lambda_i)_{i\geq 1}$ les M/B factorisations respectives de λ et Λ. On a alors pour tout $i \geq 1$:*

1. *λ_i est un bosse si et seulement si Λ_i est une bosse.*
2. *λ_i et Λ_i ont même couleur.*

Dès lors, étant donnée une partie infinie λ dans $\widetilde{\mathbb{G}}$ commençant en $(p_{in},a_{in},(T,\ldots,T),\rho(p_{in}))$ où Eve respecte φ, l'ensemble des couleurs infiniment répétées dans λ est le même que l'ensemble des couleurs infiniment répétées dans la partie Λ construite par la stratégie φ. Or, Λ est une partie gagnée par Eve puisque celle-ci respecte Φ. Dès lors, la plus petite couleur infiniment répétée dans λ est paire, et λ est donc remportée par Eve.

5.2.3 Implication réciproque

Supposons que la configuration $(p_{in},a_{in},(T,\ldots,T),\rho(p_{in}))$ soit gagnante pour Eve dans le jeu $\widetilde{\mathbb{G}}$, et considérons une stratégie positionnelle gagnante φ pour cette dernière. Nous allons définir, à partir de φ, une stratégie gagnante Φ pour Eve dans $\mathbb{G}(T)$ depuis $(p,a_{in}\bot)$.

La stratégie Φ est une stratégie à pile, c'est-à-dire qu'elle utilise comme mémoire une pile Π dont l'alphabet est l'ensemble des sommets principaux de $\widetilde{\mathcal{G}}$, c'est-à-dire $Q \times \Gamma \times \mathcal{P}(Q)^{d+1} \times \{0,\ldots,d\}$. Dans la suite, $top(\Pi)$ désignera le sommet de la pile Π. Lors d'une partie Λ où Eve respecte Φ, nous aurons à tout moment que $top(\Pi) = (p,a,\overrightarrow{R},\theta)$ si et seulement si la position courante dans Λ est de la forme (p,au) pour un certain $u \in \Gamma^*$. De plus, θ sera la plus petite couleur visitée depuis que a a été empilé. Enfin, si Eve respecte Φ et si a est un jour dépilé, la configuration alors atteinte sera de la forme (r,u), et si i désigne la plus petite couleur visitée lorsque a était dans la pile, on aura $r \in R_i$.

Au départ, la pile ne contient qu'un seul symbole, $(p_{in},a_{in},(T,\ldots,T),\rho(p_{in}))$. Supposons que nous sommes dans une configuration de la forme (p,au), et que $top(\Pi) = (p,a,\overrightarrow{R},\theta)$. Dans un premier temps, nous décrivons comment Eve joue si $p \in Q_{\mathbf{E}}$, et ensuite nous expliquons comment la pile de stratégie est mise à jour.

- **Choix du coup :** Supposons que $p \in Q_{\mathbf{E}}$ et donc qu'Eve doive jouer depuis (p,au). Pour cela, Eve considère la valeur de φ en $(p,a,\overrightarrow{R},\theta)$.
  S'il s'agit d'un coup vers $\sharp$, Eve applique une transition $pop(r)$ pour un état r tel que $r \in R_\theta$ (le lemme 5.1 montrera qu'un tel état r existe).
  Si le coup donné par φ est d'aller vers un sommet $(q,a,\overrightarrow{R},\min(\theta,\rho(q)))$, Eve applique la transition $skip(q)$.
  Si le coup donné par φ est d'aller vers un sommet $(p,a,\overrightarrow{R},\theta,q,b)$, alors Eve applique la transition $push(q,b)$.
- **Mise à jour de Π :** Si le coup, joué par Eve ou Adam, a été d'aller de (p,au) vers une configuration (r,u), Eve met à jour Π en dépilant $(p,a,\overrightarrow{R},\theta)$ et en remplaçant le nouveau sommet de pile $(q,b,\overrightarrow{S},\theta')$ par $(r,b,\overrightarrow{S},\min(\theta',\theta,\rho(r)))$. Ce cas est illustré par la figure 5.4.
  Si le coup, joué par Eve ou Adam, a consisté à aller de (p,au) vers une configuration (q,au), Eve met à jour Π en remplaçant le sommet de pile $(p,a,\overrightarrow{R},\theta)$ par $(q,a,\overrightarrow{R},\min(\theta,\rho(q)))$.
  Si le coup, joué par Eve ou Adam, a consisté à aller de (p,au) vers une configuration (q,bau), Eve met à jour Π en empilant $(q,b,\overrightarrow{S},\rho(q))$, où $\overrightarrow{S}$ est tel que $(p,a,\overrightarrow{R},\theta,q,b,\overrightarrow{S}) = \varphi(p,a,\overrightarrow{R},\theta,q,b)$.

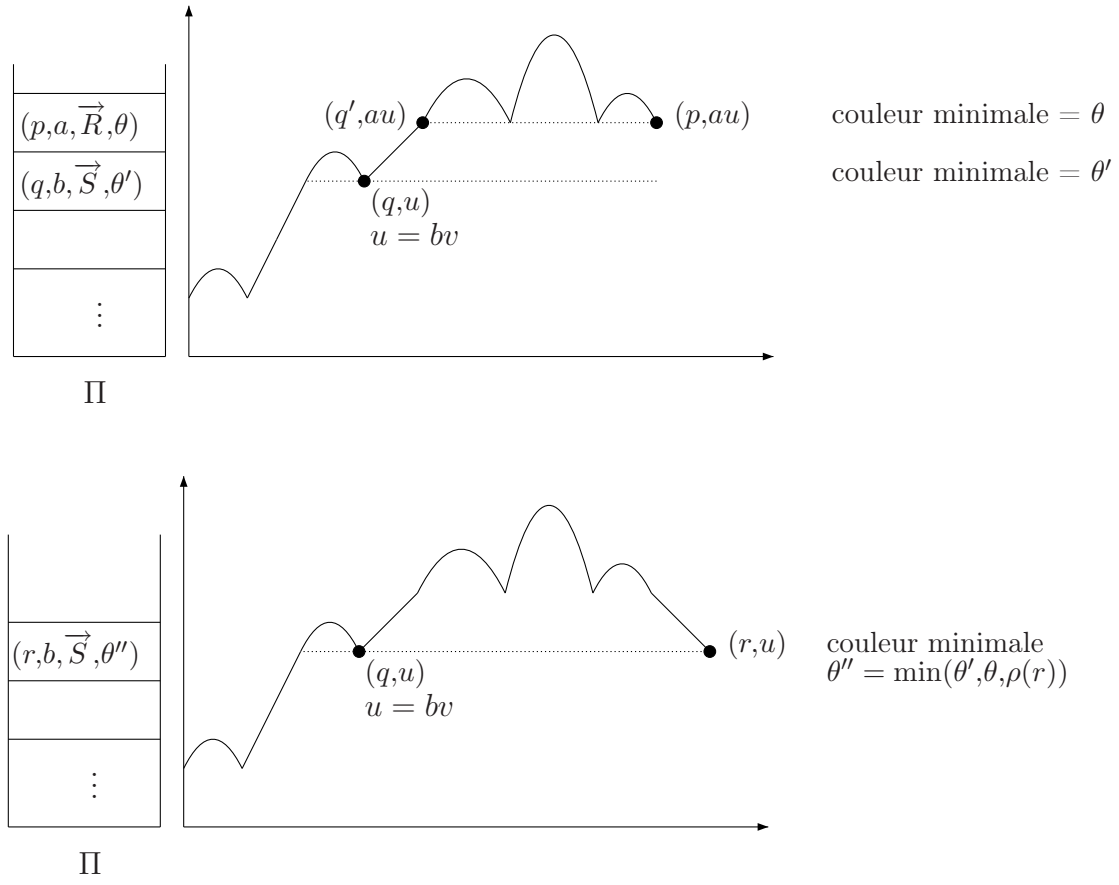


FIG. 5.4 – *Mise à jour de la pile de stratégie* Π

On peut remarquer dès maintenant que la stratégie Φ ainsi définie utilise une mémoire infinie, à savoir une pile. Cependant, Φ est finiment descriptible et à tout moment la hauteur de la pile de stratégie Π est la même que celle de la configuration courante dans le jeu. Il ne faut donc pas plus de mémoire pour stocker l'information relative Π que celle relative à la position courante.

Intuitivement, $\widetilde{\mathbb{G}}$ fournit une version compacte des parties dans le jeu $\mathbb{G}$. Pour les besoins de la preuve mais aussi pour illustrer cette intuition, on associe, à toute partie partielle Λ dans $\mathbb{G}$ commençant en $(p_{in},a_{in}\bot)$, où Eve respecte Φ, une partie $\mu(\Lambda)$ dans $\widetilde{\mathbb{G}}$ où Eve respecte φ.

La construction est par récurrence :

- au départ, c'est-à-dire quand $\Lambda = (p_{in},a_{in}\bot)$, $\mu(\Lambda) = (p_{in},a_{in},(T,\dots,T),\rho(p_{in}))$.
- supposons que pour une partie partielle Λ, on ait construit $\mu(\Lambda)$. Soit (p,au) le dernier sommet de Λ et soit $(p,a,\overrightarrow{R},\theta)$ le dernier sommet de $\mu(\Lambda)$ (le lemme 5.1 montrera qu'il a toujours cette forme). Il y a trois manières d'étendre Λ selon que l'on empile, que l'on laisse la pile inchangée ou que l'on dépile depuis (p,au) :

  - supposons que Λ est étendu par une transition $push(q,b)$. On appelle alors $\Lambda' = \Lambda \cdot (q,bau)$ l'extension de Λ et l'on pose $(p,a,\overrightarrow{R},\theta,q,b,\overrightarrow{S}) = \varphi((p,a,\overrightarrow{R},\theta,q,b))$.
    Enfin, on pose $\mu(\Lambda') = \mu(\Lambda)\cdot(p,a,\overrightarrow{R},\theta,q,b)\cdot(p,a,\overrightarrow{R},\theta,q,b,\overrightarrow{S})\cdot(q,b,\overrightarrow{S},\rho(q))$.
  - supposons que Λ est étendu par une transition $skip(q)$. On note $\Lambda' = \Lambda \cdot (q,au)$ l'extension de Λ et l'on pose $\mu(\Lambda') = \mu(\Lambda)\cdot(q,a,\overrightarrow{R},\min(\theta,\rho(q))$
  - supposons que Λ est étendu par une transition $pop(r)$. On appelle alors $\Lambda' = \Lambda \cdot (r,u)$ l'extension de Λ. La figure 5.5 illustre ce qui suit. Soit $n = |u| - 1$, et soit $\mu(\Lambda)_{|_n}$ l'unique préfixe de $\mu(\Lambda)$ se terminant juste avant la n-ème marche de $\mu(\Lambda)$ (on prouve facilement par récurrence qu'une telle occurrence existe).
    Soit $M_n = (q,b,\overrightarrow{S},\theta')(q,b,\overrightarrow{S},\theta',q',a)(q,b,\overrightarrow{S},\theta',q',a,\overrightarrow{R})(q',a,\overrightarrow{R},\rho(q'))$ la n-ième marche de $\mu(\Lambda)$.
    On pose $B = (q,b,\overrightarrow{S},\theta')(q,b,\overrightarrow{S},\theta',q',a)(q,b,\overrightarrow{S},\theta',q',a,\overrightarrow{R})(r,b,\overrightarrow{S},\theta',\theta)$ $(r,b,\overrightarrow{S},\min(\theta',\theta,\rho(r)))$. Enfin $\mu(\Lambda') = \mu(\Lambda)_{|_n}\cdot B$.

On a alors le lemme suivant qui traduit la correspondance entre Λ et $\mu(\Lambda)$

Lemme 5.1 *Soit Λ une partie partielle dans $\mathbb{G}$, où Eve respecte Φ, commençant en $(p_{in},a_{in}\bot)$ et se terminant dans une configuration (p,au). On a alors les relations suivantes entre Λ et $\mu(\Lambda)$:*

1. *$\mu(\Lambda)$ est une partie partielle dans $\widetilde{\mathbb{G}}$, où Eve respecte φ, commençant en $(p_{in},a_{in},(T,\dots,T),\rho(p_{in}))$ et se terminant en $(p,a,\overrightarrow{R},\theta)$ pour un certain $\overrightarrow{R} \in \mathcal{P}(Q)^{d+1}$ et un $\theta \in \{0,1,\dots,d\}$.*
2. *à l'issue de Λ, $Top(\Pi)$ est égal au dernier sommet de $\mu(\Lambda)$.*
3. *θ est la plus petite couleur vue depuis que a a été empilé.*

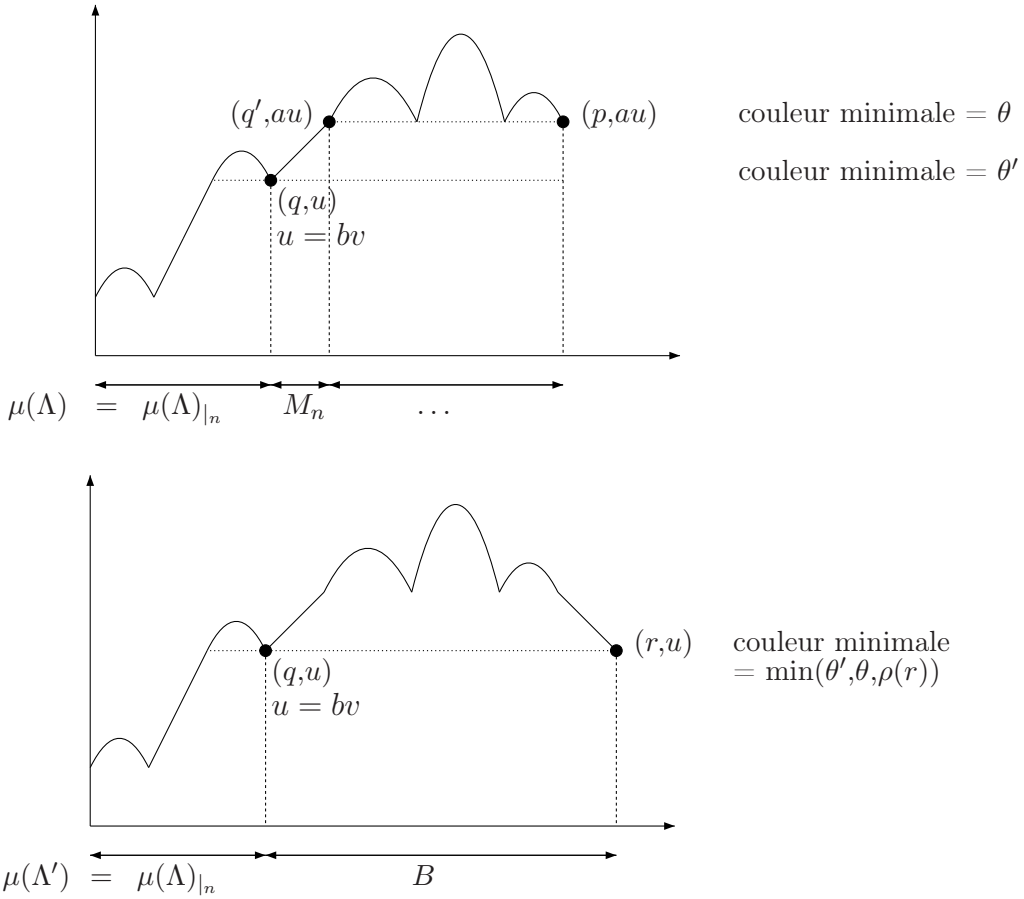


FIG. 5.5 – *Calcul de $\mu(\Lambda')$ pour une extension de type pop*

4. *si le prochain coup dans* Λ *consiste à appliquer une règle* $pop(r)$ *pour un* $r \in Q$, *alors* $r \in R_\theta$.

Preuve. La preuve est donnée par récurrence sur Λ. On commence par montrer que le dernier point est une conséquence du premier. Supposons donc que l'on prolonge Λ en appliquant une règle $pop(r)$. Dès lors, il existe dans $\widetilde{\mathcal{G}}$ un arc allant de $(p,a,\overrightarrow{R},\theta)$ vers tt ou ff, selon que l'on a $r \in R_\theta$ ou $r \notin R_\theta$. Si $p \in Q_{\mathbf{E}}$, par définition de Φ, l'arc est vers tt et donc $r \in R_\theta$. Si $p \in Q_{\mathbf{A}}$, si $r \notin R_\theta$, il existe un arc de $(p,a,\overrightarrow{R},\theta)$ vers ff, et donc Adam peut prolonger $\mu(\Lambda)$ vers ff et remporter la partie, ce qui est contradictoire avec le premier point puisque φ est un stratégie gagnante pour Eve.

Prouvons maintenant les trois premiers points. Pour cela, on suppose que le résultat est établi pour une partie Λ, et l'on considère une extension Λ' de Λ. On a alors les cas suivants :

- Λ' est obtenu à partir de Λ en appliquant une règle de type *skip*. Les trois points découlent alors des définitions précédentes.
- Λ' est obtenu à partir de Λ en appliquant une règle de type *push*. Les trois points découlent alors des définitions précédentes.
- Λ' est obtenu à partir de Λ en appliquant une règle $pop(r)$. Ce cas mérite un peu plus de détail. Tout d'abord, l'hypothèse de récurrence et le dernier point nous permettent de conclure que $r \in R_\theta$ si l'on désigne par $(p,a,\overrightarrow{R},\theta)$ le dernier sommet de $\mu(\Lambda)$. Dès lors, en reprenant la définition de $\mu(\Lambda')$ (on pourra aussi se référer à la figure 5.5 pour une vision plus graphique des arguments), on vérifie facilement que B est une bosse valide dans $\widetilde{\mathbb{G}}$ où Eve respecte sa stratégie φ. De là découle le premier point. Le deuxième ne pose alors pas de problème. Quant au troisième il vient de l'hypothèse de récurrence et du fait que $\min(\theta',\theta,\rho(r))$ est bien la plus petite couleur vue depuis que b est dans la pile, si b désigne le nouveau sommet de pile obtenu en dépilant a.

□

Le second point du lemme précédent permet d'expliciter le lien entre la pile de stratégie et $\mu(\Lambda)$, à savoir que la pile donne une version compacte de $\mu(\Lambda)$.

Lemme 5.2 *Soit* Λ *une partie partielle dans* $\mathbb{G}$, *où Eve respecte* Φ. *Soit* $Cont(\Pi)$ *le contenu de la pile de stratégie* Π *lue de bas en haut (en ignorant le symbole de fond de pile) à l'issue de* Λ.

Soit $\mu(\Lambda) = v_0v_1v_2\cdots v_k$ *Soit* $Steps_{\mu(\Lambda)} = \{n_0 < n_1 < \cdots < n_h\}$, *et soit* $Steps^{\max}_{\mu(\Lambda)} = \{n \mid \exists i \text{ t.q. } n = n_i \text{ et } v_{n_i}\cdots v_{n_{i+1}} \text{ est une marche}\} \cup \{n_h\}$.

Alors, $Cont(\Pi) = v_{m_0}v_{m_1}\cdots v_{m_l}$ *où* $\{m_0 < m_1 < \cdots < m_l\} = Steps^{\max}_{\mu(\Lambda)}$.

Preuve. Par récurrence en utilisant le deuxième point du lemme 5.1. □

Enfin, le lemme suivant donne le lien entre la M/B factorisation d'une partie partielle Λ et la M/B factorisation de $\mu(\Lambda)$

Lemme 5.3 *Soit* Λ *une partie partielle dans* $\mathbb{G}$, *où Eve respecte* Φ, *commençant en* $(q_{in},a_{in}\bot)$ *et se terminant dans une configuration* (p,au). *Soient* $(\Lambda_i)_{i=0\ldots h}$ *et* $(\lambda_i)_{i=0\ldots k}$ *les M/B factorisations respectives de* Λ *et* $\mu(\Lambda)$.

- $h = k$.

– *pour tout* $i = 0 \dots h$, Λ_i *et* λ_i *ont même nature (bosse ou marche) et même couleur.*

Preuve. Le premier point ne pose pas de problème. Pour le second point, le fait que Λ_i et λ_i ait même nature vient de la définition. La propriété sur la couleur est facile dans le cas des marches et des bosses triviales. Pour ce qui est du cas général des bosses, c'est une conséquence facile du troisième point du lemme 5.1 et de la définition de $\mu(\Lambda)$. □

Considérons maintenant une partie infinie Λ dans $\mathbb{G}$ commençant en $(q_{in},a_{in}\bot)$, et où Eve respecte Φ. Les lemmes 5.1 et 5.3 s'appliquent aux parties partielles. Une version sur les parties infinies permettrait de conclure que Λ est gagnante en considérant une version infinie de $\mu(\Lambda)$ (où Eve respecterait φ et qui serait donc gagnante) et les lemmes précédents. Un tel passage à la limite est possible à condition de définir la version infinie de $\mu(\Lambda)$.

Pour tout entier i, appelons $\Lambda^i = \Lambda\restriction_i$ le préfixe de longueur i de Λ, et notons $\lambda^i = \mu(\Lambda^i)$. Il y a deux comportement possibles pour Λ. Soit un niveau de pile est infiniment répété, soit la pile explose strictement. Dans les deux cas, on vérifie facilement que pour tout entier k, les k premiers sommets des λ^i sont les mêmes pour tout i plus grand qu'une borne j_k, c'est-à-dire que $\lambda^i\restriction_k = \lambda^{j_k}\restriction_k$ pour tout $i \geq j_k$. On prend enfin pour $\mu(\Lambda)$ le mot infini, tel que pour tout $k \geq 0$, $\mu(\Lambda)\restriction_k = \lambda^{j_k}\restriction_k$

Dès lors, avec cette définition de $\mu(\Lambda)$, on obtient une variante des lemmes 5.1 et 5.3 pour les partie infinies. Comme $\mu(\Lambda)$ est une partie où Eve suit sa stratégie gagnante φ, celle-ci est gagnante et dès lors la plus petite couleur apparaissant infiniment souvent dans la M/B factorisation de $\mu(\Lambda)$ est paire. Dès lors, il en est de même pour Λ, ce qui permet de conclure que Φ est une stratégie gagnante.

5.2.4 Un exemple complet

Tout au long du chapitre 4, nous avions étudié un exemple de jeu de parité sur un graphe de processus à pile. En particulier, nous avions calculé *à la main* les ensembles de retours et proposé une description d'une stratégie gagnante au départ d'un sommet particulier.

Nous reprenons le même exemple ici et utilisons la réduction précédente pour calculer les ensembles de retours et illustrer la stratégie à pile proposée dans la preuve de la réciproque du théorème 5.1.

On considère donc le processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ donné dans l'exemple 4.3 :

– $Q = \{p,q,r\}$: $Q_{\mathbf{E}} = \{p,q\}$ et $Q_{\mathbf{A}} = \{r\}$.
– $\Gamma = \{a,b,\bot\}$.
– la relation Δ est donnée par :
  – $\Delta(p,a) = \{pop(q),push(r,a)\}$.
  – $\Delta(q,a) = \{pop(p),push(r,a)\}$.
  – $\Delta(r,a) = \{pop(p),pop(r)\}$.
  – $\Delta(p,b) = \{pop(p),push(p,a)\}$.
  – $\Delta(q,b) = \{pop(p),push(r,a)\}$.
  – $\Delta(r,b) = \{pop(p),push(q,a)\}$.

  - $\Delta(p,\bot) = \{push(p,a),push(r,b)\}$.
  - $\Delta(q,\bot) = \{push(q,a),push(r,a)\}$.
  - $\Delta(r,\bot) = \{push(q,b),push(r,a)\}$.
- $\rho(p) = 0$, $\rho(q) = 2$ et $\rho(r) = 1$.

Nous avions déterminé (exemple 4.4) les ensembles de retours suivants ainsi que la nature des positions de pile vide :

$$\begin{array}{lcllcl} \mathcal{R}(p,a) & = & \{\{q\},\{p,r\}\} & \mathcal{R}(p,b) & = & \{\{p\}\} \\ \mathcal{R}(q,a) & = & \{\{p\}\} & \mathcal{R}(q,b) & = & \{\{p\}\} \\ \mathcal{R}(r,a) & = & \{\{p,r\}\} & \mathcal{R}(r,b) & = & \{\{p\}\} \end{array}$$

$(p,\bot) \in W_{\mathbf{E}}$ $\quad$ $(q,\bot) \in W_{\mathbf{E}}$ $\quad$ $(r,\bot) \in W_{\mathbf{A}}$

Nous expliquons maintenant comment trouver ces résultats formellement à l'aide de la réduction vers le jeu $\widetilde{\mathbb{G}}$ décrit ci-dessus. Il serait trop long de montrer la minimalité des ensembles de retours, mais nous explicitons les stratégie sous-jacentes dans le graphe $\widetilde{\mathcal{G}}$. Par exemple, nous montrons que $\{p\}$ est dans $\mathcal{R}(r,b)$, mais nous ne montrons pas qu'il est minimal. Nous décrivons en revanche la stratégie gagnante pour Eve depuis $(r,b,(\{p\},\{p\},\{p\}),1)$ dans $\widetilde{\mathcal{G}}$.

Pour cela, nous allons décrire la structure locale de $\widetilde{\mathcal{G}}$ autour des sommets concernés. Nous ne représenterons pas toutes les arcs partant des sommets d'Eve, mais seulement ceux qui correspondent à des coups donnés par une stratégie gagnante. Cela permet alors de vérifier le caractère gagnant d'une position. Nous ne représentons pas non plus les sommets intermédiaires de la forme $(s,a,\overrightarrow{R},\theta,i)$ utilisés dans les bosses mais nous les remplaçons par un arc direct coloré par i d'un sommet de la forme $(p,a,\overrightarrow{R},\theta,q,b,\overrightarrow{S})$ vers un sommet $(s,a,\overrightarrow{R},\theta)$ avec $s \in S_i$.

Pour ce qui est des ensembles de retours, la figure 5.6 synthétise les résultats. Les seules positions pour lesquelles on ne doit pas simuler un dépilement sont $(r,b,(\{p\},\{p\},\{p\}),1)$ et $(p,a,(\{p,r\},\{p,r\},\{p,r\}),0)$.

La figure 5.7 montre que les configurations $(p,\bot)$ et $(q,\bot)$ sont gagnantes pour Eve.

Illustrons enfin le calcul d'une stratégie pour Eve dans le jeu $\mathbb{G}$ à l'aide d'une stratégie dans $\widetilde{\mathbb{G}}$. On considère la configuration $(r,b\bot)$ et le jeu conditionnel $\mathbb{G}(\{p\})$. Au départ, la pile de stratégie contient seulement le sommet $(r,b,(\{p\},\{p\},\{p\}),1)$. Imaginons qu'Adam aille depuis $(r,b\bot)$ en $(q,ab\bot)$. Eve met alors à jour la pile de stratégie en considérant la valeur de sa stratégie gagnante dans $\widetilde{\mathbb{G}}$ depuis $(r,b,(\{p\},\{p\},\{p\}),1,q,a)$, à savoir $(r,b,(\{p\},\{p\},\{p\}),1,q,a,(\emptyset,\emptyset,\{p\}))$. Elle empile alors dans la pile de stratégie $(q,a,(\emptyset,\emptyset,\{p\}),2)$. En $(q,ab\bot)$, Eve doit jouer. Elle considère donc le coup donné dans $\widetilde{\mathbb{G}}$ depuis le sommet de sa pile de stratégie $(q,a,(\emptyset,\emptyset,\{p\}),2)$. Comme il s'agit d'un coup vers $\#$ induit par la règle $pop(p)$, elle applique cette règle et va donc dans $(p,b\bot)$. Pour la mise à jour de sa pile de stratégie, elle supprime le sommet de pile et remplace le nouveau sommet $(r,b,(\{p\},\{p\},\{p\}),1)$ par $(p,b,(\{p\},\{p\},\{p\}),0)$ (elle actualise l'état de contrôle ainsi que la dernière composante). Depuis la configuration $(p,b\bot)$, Eve doit à nouveau jouer. Elle regarde la valeur de la stratégie dans $\widetilde{\mathbb{G}}$ depuis le sommet de sa pile de stratégie $(p,b,(\{p\},\{p\},\{p\}),0)$. Comme il s'agit d'un

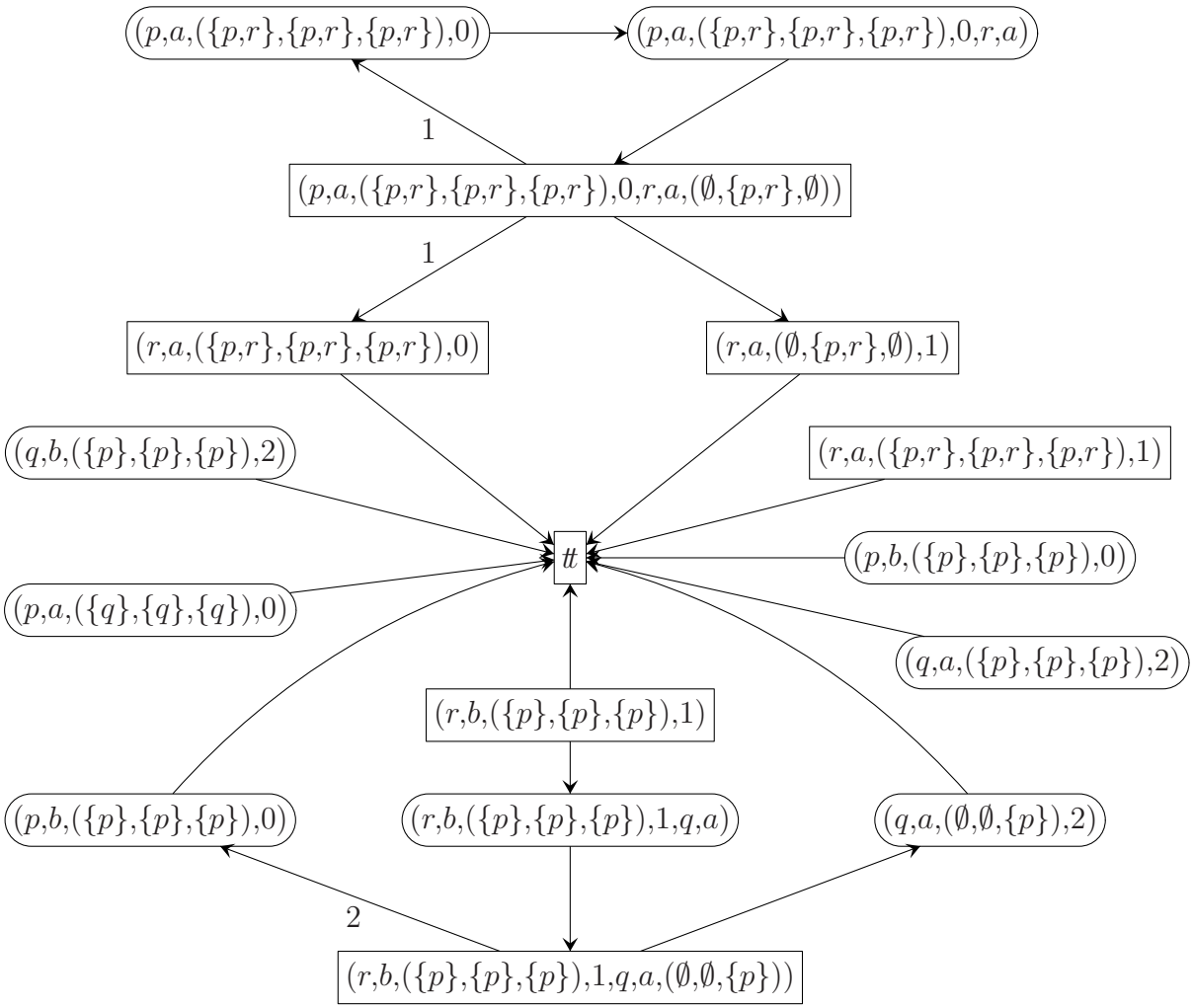


FIG. 5.6 – *Ensembles de retours*

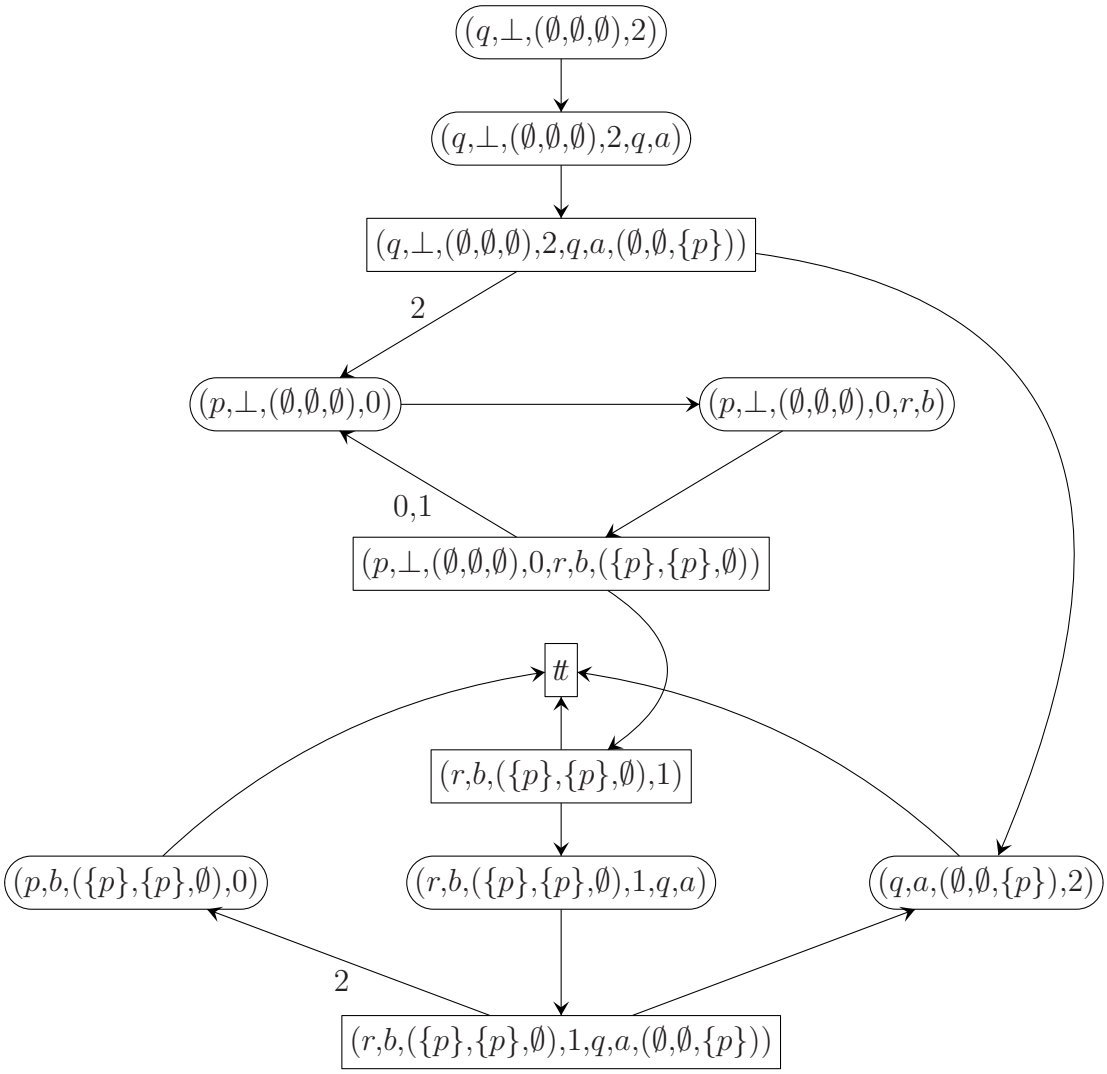


FIG. 5.7 – *Structure de* $\widetilde{G}$ *autour de* $(p,\bot,(\emptyset,\emptyset,\emptyset),0)$ *et* $(q,\bot,(\emptyset,\emptyset,\emptyset),2)$

coup vers $\#\!\!\#$ induit par $pop(p)$, elle applique cette règle. La partie s'achève alors, puisque l'on a atteint la configuration de pile vide $(p,\perp)$.

5.2.5 A propos de la taille du graphe fini

Le graphe $\widetilde{\mathcal{G}}$ est de taille exponentielle dans la taille de $\mathcal{P}$. Nous donnons ici une expression exacte du nombre de sommets qui le composent. Pour cela, on note $n = |Q|$, $d = |C|$ et $m = |\Gamma|$. On a alors $nm2^{nd}d$ sommets principaux, $nm2^{nd}dn(m-1)$ sommets à six composantes, $nm2^{nd}dn(m-1)2^{nd}$ sommets à sept composantes, $n(m-1)2^{nd}d^2$ sommets à cinq composantes, auxquels il faut ajouter les deux sommet $\#\!\!\#$ et fff. Au total, le nombre de sommets de $\widetilde{\mathcal{G}}$ est donc :

$$n2^{nd}d\left[(m-1)d^2 + m(1+n(m-1)(1+2^{nd}))\right] + 2$$

Dans le cas de l'exemple précédent, on obtient donc un graphe avec 42 647 042 sommets! Cet exemple montre les limites de la méthode précédente si l'on se place dans un cadre pratique. En effet, un tel graphe est trop grand pour une implémentation de l'algorithme de calcul des ensembles de retours. Cependant, nous venons de le voir, dans le paragraphe 5.2.4, beaucoup de sommets sont en fait inutiles. Il serait donc intéressant (mais nous ne le ferons pas ici), dans l'optique d'une implémentation, de proposer un algorithme de réduction du nombre de sommets du graphe $\widetilde{\mathcal{G}}$.

5.3 Conditions d'explosion stricte

5.3.1 La réduction

Dans la proposition 5.1, nous avions noté que dans un jeu muni d'une condition d'explosion stricte, une partie est remportée par Eve si et seulement si sa M/B factorisation contient une infinité de marches, ce qui traduit que tout niveau de pile est quitté un jour pour toujours.

Tout naturellement, on reprend le graphe de jeu $\widetilde{\mathcal{G}}$ introduit dans le paragraphe 5.2, et on le simplifie de la façon suivante :

- tout d'abord, on supprime le coloriage qui n'a plus de sens ici.
- pour les états atteints en dépilant, on n'a plus besoin d'un vecteur de parties de Q, mais seulement d'une partie de Q.
- la composante θ ne servant plus à rien, on la supprime.
- les conditions pour aller vers $\#\!\!\#$ ou fff sont simplifiées : elles regardent si l'état atteint en dépilant est dans l'ensemble R ou non.
- enfin, les sommets de la forme $(s,a,\overrightarrow{R},\theta,i)$ ne sont plus utiles puisque leur seul intérêt était de traduire la couleur d'une bosse.

On ajoute en revanche un sommet intermédiaire dans la simulation d'une marche que l'on marque comme final, et l'on considère alors une condition de Büchi sur ce nouveau graphe (on visite une infinité d'état finaux si et seulement si l'on simule une infinité de marches).

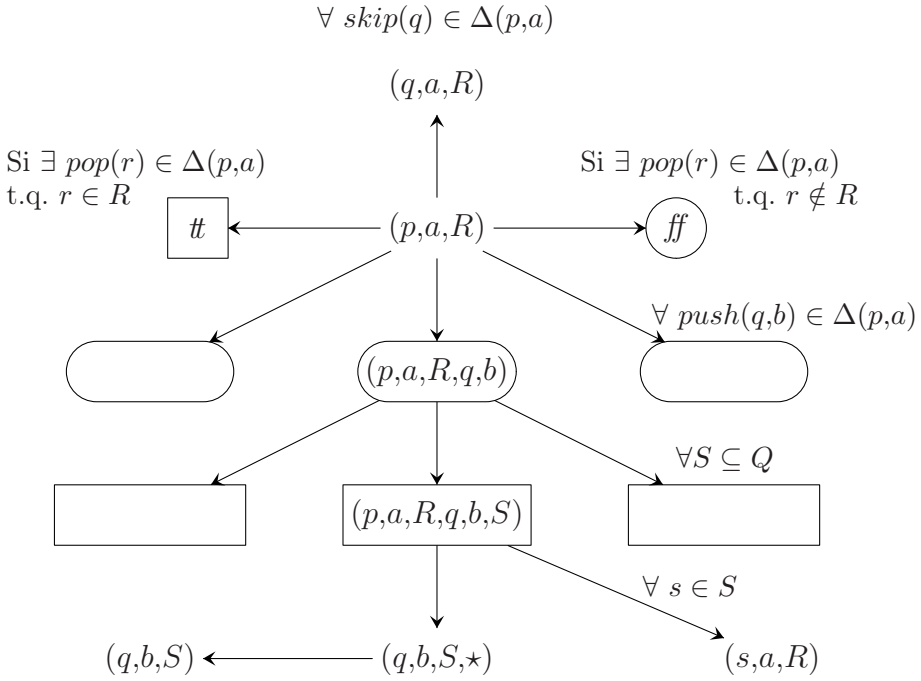


FIG. 5.8 – *Structure locale de $\widetilde{\mathcal{G}}$ pour la condition d'explosion stricte.*

Le graphe obtenu est donné dans la figure 5.8. Enfin, on note $\widetilde{\mathbb{G}}$ le jeu de Büchi sur $\widetilde{\mathcal{G}}$.

On a alors le résultat suivant, qui est l'analogue du théorème 5.1.

Théorème 5.2 *Soit $p_{in} \in Q$, soit $a_{in} \in \Gamma$ et soit $T \subseteq Q$. On a les équivalences suivantes :*

- *Eve possède une stratégie gagnante dans $\mathbb{G}(T)$ depuis $(p_{in},a_{in}\bot)$ si et seulement si elle possède une stratégie gagnante dans $\widetilde{\mathbb{G}}$ depuis (p_{in},a_{in},T).*
- *Eve possède une stratégie gagnante dans $\mathbb{G}$ depuis $(p_{in},\bot)$ si et seulement si elle possède une stratégie gagnante dans $\widetilde{\mathbb{G}}$ depuis $(p_{in},\bot,\emptyset)$.*

De plus, on peut construire une stratégie gagnante à pile dans $\mathbb{G}$ pour Eve depuis toute position gagnante.

5.3.2 A propos de la preuve

Nous ne donnons pas ici la preuve du théorème 5.2, puisque celle-ci est très proche de celle du théorème 5.1. De plus, le théorème 5.2 sera un corollaire du théorème 6.6 comme expliqué dans le paragraphe 6.3.2. Nous nous contentons donc de donner la trame en signalant les changements par rapport à la preuve du théorème 5.1.

Tout d'abord, on définit la notion de M/B factorisation dans $\widetilde{\mathcal{G}}$ en bosses et en marches. Pour l'implication directe, la stratégie est définie de la même façon que précédemment, sauf que le calcul de l'ensemble des états pouvant être atteints en dépilant est plus simple, puisque l'on ne doit pas raffiner celui-ci en un vecteur

(il n'y a pas de couleur). La stratégie au cours d'une partie λ dans $\widetilde{\mathbb{G}}$, construit comme précédemment une partie Λ dans $\mathbb{G}$. La preuve se termine en montrant la correspondance entre les M/B factorisations de λ et Λ, et en remarquant que, puisque celle de Λ contient une infinité de marches, il en est de même pour celle de λ, et donc que λ vérifie la condition de Büchi (une marche contient un état final).

Pour l'implication réciproque, on utilise à nouveau une pile de stratégie Π, dans laquelle est stockée une version compacte d'une partie dans $\widetilde{\mathbb{G}}$. Le choix du coup dans $\mathbb{G}$ est fait à l'aide de la valeur de la stratégie dans $\widetilde{\mathbb{G}}$ sur le sommet de la pile de stratégie. Pour les besoins de la preuve, on associe à toute partie Λ dans $\mathbb{G}$ une partie $\mu(\Lambda)$ où Eve respecte sa stratégie, et qui peut être vue comme une décompression de la pile de stratégie Π. La preuve se termine en montrant la correspondance entre les M/B factorisations de Λ et $\mu(\Lambda)$. En remarquant que $\mu(\Lambda)$ vérifie la condition de Büchi, on en déduit que sa M/B factorisation contient une infinité de marches (seules les marches contiennent un état final), et il en est donc de même pour celle de Λ.

5.3.3 A propos de la taille du graphe fini

Remarquons que le graphe est toujours exponentiel (mais plus dans le nombre de couleurs puisqu'il n'y en a pas). Si l'on note $n = |Q|$ et $m = |\Gamma|$, le nombre de sommets de $\widetilde{\mathcal{G}}$ est :

$$nm2^n\left[2 + n(m-1)(1+2^n)\right] + 2$$

Contrairement au cas de la condition de parité, le graphe de jeu $\widetilde{\mathcal{G}}$ est cette fois plus petit, et le jeu $\widetilde{\mathbb{G}}$ associé est muni d'une condition de gain plus simple, à savoir une condition de Büchi. Une implémentation efficace paraît alors bien plus réalisable.

5.4 De l'explosion stricte à l'explosion

Considérons maintenant une condition d'explosion : Eve gagne une partie si et seulement si la pile n'est pas bornée au cours de la partie. On commence par remarquer qu'il existe des stratégies positionnelles pour Eve dans un tel jeu.

Lemme 5.4 *Soit $\mathbb{G}$ un jeu muni d'une condition d'explosion. Si Eve possède une stratégie gagnante depuis une position v, alors elle possède une stratégie gagnante positionnelle depuis v.*

Preuve. Appelons $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ le processus à pile sous-jacent et $\mathcal{G}$ le graphe de jeu, et considérons une stratégie gagnante φ pour Eve dans $\mathbb{G}$ depuis v. En particulier, pour tout $i \geq 1$, φ est une stratégie gagnante dans le jeu d'accessibilité $\mathbb{G}_i$ vers l'ensemble $Q \times \Gamma^{\geq i}$ des configurations de hauteur de pile supérieure ou égale à i.

Pour tout $i \geq 1$, la configuration v étant gagnante dans le jeu d'accessibilité $\mathbb{G}_i$, Eve possède une stratégie positionnelle ψ_i depuis v dans ce jeu. De plus, il est clair que ψ_i est une stratégie positionnelle gagnante pour Eve depuis v dans les jeux $\mathbb{G}_j$ pour $j \leq i$.

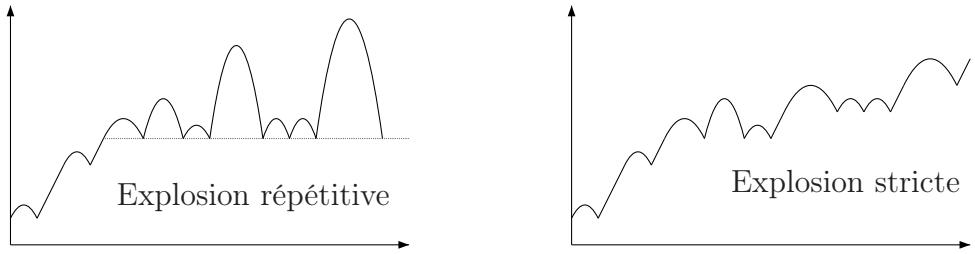


FIG. 5.9 – *Les deux types de partie gagnantes pour l'explosion*

A partir de la suite $(\psi_i)_{i\geq 1}$, il n'est pas difficile de construire une stratégie positionnelle gagnante dans $\mathbb{G}$ depuis v. Pour cela on ordonne les sommets de $\mathcal{G}$ par taille de pile croissante : on obtient alors une suite $(v_j)_{j\geq 0}$, telle que $\{v_j \mid j \in \mathbb{N}\} = Q \times \Gamma^*$ (les premiers sommets sont ceux de pile vide, puis viennent ceux de hauteur de pile égale à 1 et ainsi de suite). Appelons enfin $I_0 = \mathbb{N}$.

Comme v_0 n'a qu'un nombre fini de successeurs possibles, il en possède un, w_0, tel que $\psi_i(v_0) = w_0$ pour une infinité d'indices $i \in I_0$. Appelons I_1 cet ensemble infini d'indices et posons $\psi(v_0) = w_0$. Comme v_1 n'a qu'un nombre fini de successeurs possibles, il en possède un, w_1, tel que $\psi_i(v_1) = w_1$ pour une infinité d'indices $i \in I_1$. Appelons I_2 cet ensemble infini d'indices et posons $\psi(v_1) = w_1$.

En raisonnant de la sorte, on définit une stratégie positionnelle ψ, ainsi qu'une suite infinie décroissante d'ensembles infinis d'indices $(I_i)_{i\geq 0}$, tels que pour tout entier k, $\psi(v_j) = \psi_i(v_j)$ pour tout $0 \leq j \leq k$ et tout $i \in I_k$.

Par l'absurde, supposons que ψ soit une stratégie perdante pour Eve dans $\mathbb{G}$ depuis v. Dès lors, il existe une partie Λ dans $\mathbb{G}$ où Eve respecte ψ et dans lequel la pile est bornée par un entier N. Appelons k le plus petit indice tel que v_k soit de hauteur de pile égale à $N+1$. En d'autres termes, $\{v_j \mid j < k\} = Q \times \Gamma^{\leq N}$. Considérons enfin un entier i dans I_k tel que $i \geq N+1$ (un tel entier existe puisque I_k est infini). Comme $\psi(v_j) = \psi_i(v_j)$ pour tout $0 \leq j \leq k$ et comme toutes les configurations de Λ appartiennent à $\{v_j \mid j < k\} = Q \times \Gamma^{\leq N}$, Λ peut être vue comme une partie dans laquelle Eve respecte sa stratégie positionnelle ψ_i. Mais, comme ψ_i est gagnante dans le jeu d'accessibilité vers $Q \times \Gamma^{\geq i}$, Λ atteint un jour une configuration de hauteur de pile supérieure ou égale à $i \geq N+1$, ce qui est contradictoire avec le fait que la hauteur de pile soit bornée par N dans Λ.

Dès lors, on en conclut donc que ψ est une stratégie positionnelle gagnante pour Eve depuis v dans $\mathbb{G}$. □

Dans un jeu d'explosion, il y a, a priori, deux sortes de parties gagnantes pour Eve (voir figure 5.9) : celle où la pile explose strictement, et celle où elle répète infiniment un niveau tout en étant non bornée (on parle alors d'*explosion répétitive*). En fait, Eve possédant une stratégie positionnelle depuis toute position gagnante dans le jeu d'explosion, elle peut faire en sorte de ne jamais passer deux fois par la même configuration dans une partie gagnante et ainsi faire exploser strictement la pile. Plus précisément, on a le corollaire suivant du lemme 5.4 :

Corollaire 5.1 *Soit $\mathcal{G}$ un graphe de processus à pile. Soient $\mathbb{G}_{Exp}$ et $\mathbb{G}_{ExpStr}$ les*

jeux sur $\mathcal{G}$ équipés respectivement de la condition d'explosion et de la condition d'explosion stricte. Une position est gagnante pour Eve dans $\mathbb{G}_{Exp}$ si et seulement si elle est gagnante dans $\mathbb{G}_{ExpStr}$.

En particulier, on peut construire une stratégie gagnante à pile dans $\mathbb{G}_{Exp}$ pour Eve depuis toute position gagnante.

5.5 Condition de répétition et de bornage

Considérons maintenant le jeu $\mathbb{G}$ de répétition sur $\mathcal{G}$: Eve gagne si et seulement si un niveau de pile est infiniment souvent répété. La condition duale pour Adam étant la condition d'explosion stricte, si l'on veut décider si une position est gagnante pour Eve dans $\mathbb{G}$, il suffit de décider si cette position est gagnante pour Adam dans le jeu d'explosion stricte sur $\mathcal{G}$.

Le reproche que l'on peut faire à cette solution est que l'on ne peut pas en déduire une stratégie à pile gagnante pour Eve. On reprend alors le graphe de jeu fini $\widetilde{\mathcal{G}}$ donné pour la condition d'explosion stricte (voir figure 5.8) et l'on considère le jeu $\widetilde{\mathbb{G}}$ de co-Büchi sur $\mathcal{G}$. La condition de co-Büchi exprime le fait que l'on veut qu'il n'y ait qu'un nombre fini de marches, c'est-à-dire qu'un niveau soit infiniment souvent répété. On a alors le résultat suivant.

Théorème 5.3 *Soit $p_{in} \in Q$, soit $a_{in} \in \Gamma$ et soit $T \subseteq Q$. On a les équivalences suivantes :*

- *Eve possède une stratégie gagnante dans $\mathbb{G}(T)$ depuis $(p_{in},a_{in}\bot)$ si et seulement si elle possède une stratégie gagnante dans $\widetilde{\mathbb{G}}$ depuis (p_{in},a_{in},T).*
- *Eve possède une stratégie gagnante dans $\mathbb{G}$ depuis $(p_{in},\bot)$ si et seulement si elle possède une stratégie gagnante dans $\widetilde{\mathbb{G}}$ depuis $(p_{in},\bot,\emptyset)$.*

De plus, on peut construire une stratégie gagnante à pile dans $\mathbb{G}$ pour Eve depuis toute position gagnante.

Concernant la condition de bornage (Eve gagne si et seulement si la pile est bornée), on a la version duale du corollaire 5.1

Corollaire 5.2 *Soit $\mathcal{G}$ un graphe de processus à pile. Soient $\mathbb{G}_{Bor}$ et $\mathbb{G}_{Rep}$ les jeux sur $\mathcal{G}$ équipés respectivement de la condition de bornage et de la condition de répétition. Une position est gagnante pour Eve dans $\mathbb{G}_{Bor}$ si et seulement si elle est gagnante dans $\mathbb{G}_{Rep}$.*

En particulier, on peut construire une stratégie gagnante à pile dans $\mathbb{G}_{Bor}$ pour Eve depuis toute position gagnante.

5.6 Condition de parité en escalier

5.6.1 La réduction

Dans la proposition 5.1, nous avions noté que dans un jeu muni d'une condition de parité en escalier, une partie est remportée par Eve si et seulement si la suite

des couleurs des premiers sommets des facteurs de la M/B factorisation vérifie la condition de parité.

Tout naturellement, on reprend le graphe de jeu $\widetilde{\mathcal{G}}$ introduit dans le paragraphe 5.2, et on le simplifie de la façon suivante :

- pour les états atteints en dépilant, on n'a plus besoin d'un vecteur de parties de Q, mais seulement d'une partie de Q.
- la composante θ ne servant plus à rien, on la supprime.
- les conditions pour aller vers tt ou ff sont simplifiées : elles regardent si l'état atteint en dépilant est dans l'ensemble R ou non.
- enfin, les sommets de la forme $(s,a,\overrightarrow{R},\theta,i)$ ne sont plus utiles, puisque leur seul intérêt était de traduire la couleur d'une bosse.
- les seuls sommets colorés sont donc maintenant ceux de la forme (p,a,R), et ils correspondent bien aux premiers sommets des rounds. La couleur d'un sommet (p,a,R) est toujours $\rho(p)$.

Le graphe obtenu est donné dans la figure 5.10. Enfin, on note $\widetilde{\mathbb{G}}$ le jeu de parité sur $\widetilde{\mathcal{G}}$.

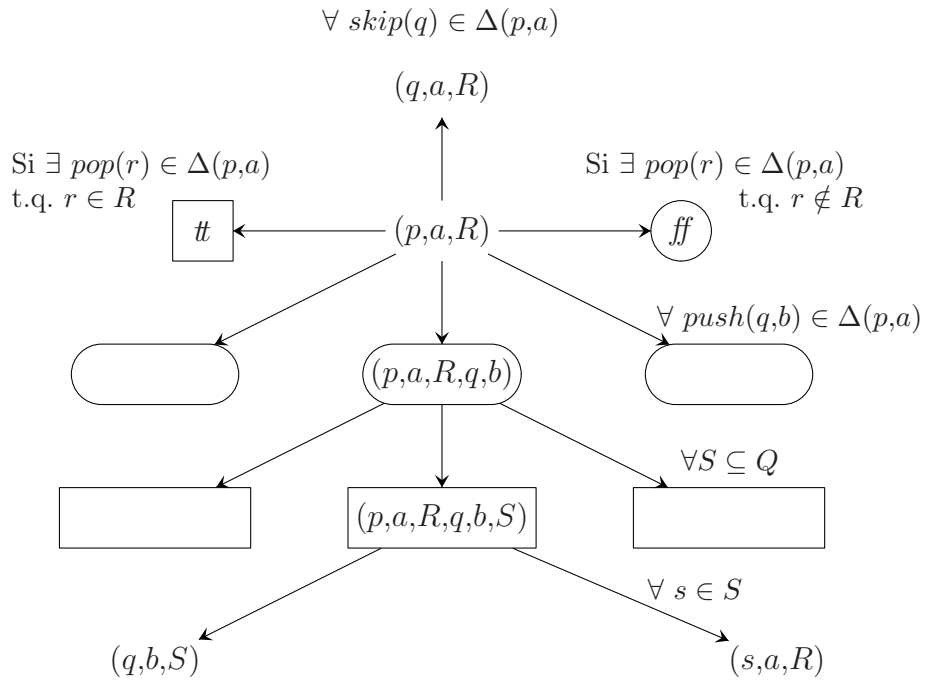


FIG. 5.10 – *Structure locale de $\widetilde{\mathcal{G}}$ pour la condition de parité en escalier.*

On a alors le résultat suivant, qui est l'analogue du théorème 5.1.

Théorème 5.4 *Soit $p_{in} \in Q$, soit $a_{in} \in \Gamma$ et soit $T \subseteq Q$. On a les équivalences suivantes :*

- *Eve possède une stratégie gagnante dans $\mathbb{G}(T)$ depuis $(p_{in},a_{in}\bot)$ si et seulement si elle possède une stratégie gagnante dans $\widetilde{\mathbb{G}}$ depuis (p_{in},a_{in},T).*

– *Eve possède une stratégie gagnante dans* $\mathbb{G}$ *depuis* $(p_{in},\bot)$ *si et seulement si elle possède une stratégie gagnante dans* $\widetilde{\mathbb{G}}$ *depuis* $(p_{in},\bot,\emptyset)$.

De plus, on peut construire une stratégie gagnante à pile dans $\mathbb{G}$ *pour Eve depuis toute position gagnante.*

5.6.2 A propos de la preuve

Nous ne donnons pas ici la preuve du théorème 5.4 puisque celle-ci est très proche de celle du théorème 5.1. Nous nous contentons donc de donner la trame en signalant les changements par rapport à la preuve du théorème 5.1. On trouvera la preuve complète de ce résultat dans [56].

Tout d'abord, on définit la notion de M/B factorisation dans $\widetilde{\mathcal{G}}$ en bosses et en marches. Pour l'implication directe, la stratégie est définie de la même façon que précédemment sauf que le calcul de l'ensemble des états pouvant être atteints en dépilant est plus simple puisque l'on ne doit pas raffiner celui-ci en un vecteur (nous ne sommes pas intéressés par la couleur d'une bosse). La stratégie au cours d'une partie λ dans $\widetilde{\mathbb{G}}$ construit comme précédemment une partie Λ dans $\mathbb{G}$. La preuve se termine en montrant la correspondance entre les M/B factorisations de λ et Λ, en particulier en montrant que les premiers sommets de chaque facteur ont la même couleur. On conclut alors en notant que Λ est gagnante pour Eve.

Pour l'implication réciproque, on utilise à nouveau une pile de stratégie Π dans laquelle est stockée une version compacte d'une partie dans $\widetilde{\mathbb{G}}$. Le choix du coup dans $\mathbb{G}$ est fait à l'aide de la valeur de la stratégie dans $\widetilde{\mathbb{G}}$ sur le sommet de la pile de stratégie. Pour les besoins de la preuve, on associe à toute partie Λ dans $\mathbb{G}$ une partie $\mu(\Lambda)$ où Eve respecte sa stratégie, et qui peut être vue comme une décompression de la pile de stratégie Π. La preuve se termine en montrant la correspondance entre les M/B factorisations de Λ et $\mu(\Lambda)$, en particulier en notant que les premiers sommets de chaque facteur ont la même couleur. En remarquant que $\mu(\Lambda)$ vérifie la condition de parité, on en conclut que la suite des couleurs des premiers sommets de sa M/B factorisation vérifie aussi la condition de parité (ce sont les seuls sommets colorés). Il en est donc de même pour la M/B factorisation de Λ.

5.6.3 A propos de la taille du graphe fini

Remarquons que le graphe est toujours exponentiel, mais plus dans le nombre de couleurs. Si l'on note $n = |Q|$ et $m = |\Gamma|$, le nombre de sommets de $\widetilde{\mathcal{G}}$ est :

$$nm2^n\left[1 + n(m-1)(1+2^n)\right] + 2$$

Contrairement au cas de la condition de parité, le graphe de jeu $\widetilde{\mathcal{G}}$ est cette fois plus petit. Cependant, la condition de gain dans $\widetilde{\mathbb{G}}$ est une condition de parité, et dès lors, une réduction du nombre de sommets dans $\widetilde{\mathcal{G}}$ serait très utile en vue d'une implémentation.

5.6.4 A propos des stratégies

L'existence de stratégies positionnelles pour les jeux équipés d'une condition de parité est un résultat classique [30, 87]. Il est alors naturel de se demander s'il en est de même, dans le cadre des jeux sur un graphe de processus à pile, pour une condition de parité en escalier. Dans la preuve du théorème 5.4, on construit une stratégie pour de tels jeux, utilisant une pile comme mémoire. Cette pile de stratégie a la même hauteur que celle du sommet courant de la partie. La proposition suivante montre l'optimalité de cette stratégie.

Proposition 5.5 *Il existe un jeu sur un graphe de processus à pile muni d'une condition de parité en escalier, et une position gagnante pour Eve dans ce jeu tels que toute stratégie gagnante pour Eve nécessite une mémoire infinie.*

Preuve. Commençons par une description informelle du jeu. Les lettres se trouvant dans la pile ne jouent pas ici un rôle très important. Adam empile des lettres tandis qu'Eve dépile. Dans une position d'Adam, celui-ci peut empiler ou décider de passer la main à Eve. En particulier il peut décider qu'Eve ne jouera pas dans toute la partie (s'il ne fait qu'empiler). Symétriquement, lorsque c'est à Eve de jouer, cette dernière peut dépiler ou passer la main à Adam. Elle peut être bloquée si elle a dépilé trop de lettres (le sommet de pile est alors $\perp$ ou b).

Au départ, c'est Adam qui joue et l'état de contrôle, p_4, est coloré par 4 : il peut rester dans cet état (et empiler un a) ou changer pour un état, p_2, coloré par 2 (et empiler un a). En p_2, il est forcé d'empiler un a et de passer dans l'état p_1 qui est coloré par 1. En p_1, il est forcé d'empiler un a et de passer dans l'état p'_4, qui est coloré par 4. Enfin, en p'_4, il empile un a et peut, soit rester dans le même état, soit aller dans l'état p_3.

L'état p_3 appartient à Eve, et cette dernière peut dépiler un a et rester dans le même état, ou empiler un b et aller dans l'état p_4 de départ.

Ainsi, une partie (où Adam ne garde pas la main indéfiniment) commence par une série d'empilements d'Adam qui passe par un état coloré par 1. Eve dépile ensuite des symboles jusqu'à passer la main à Adam. De plus, une fois la main passée, les symboles qui n'ont pas été dépilés ne le seront jamais (à cause du b). Ainsi, si Eve veut gagner, elle doit faire en sorte que 1 n'apparaisse pas dans la suite des couleurs sur laquelle est évaluée la condition de parité. Pour cela, elle doit dépiler plus de a qu'Adam en a empilé depuis son passage en p_1. Par ailleurs, si elle en dépile plus, le 2 va aussi disparaître de la suite des couleurs, et la plus petite couleur sera 3. Ainsi, une stratégie gagnante pour Eve consiste à enlever exactement les a qu'Adam a empilé depuis le passage en p_1. Ce nombre étant arbitraire, une mémoire finie ne peut suffire pour gagner.

Nous commençons par donner une stratégie gagnante pour Eve depuis $(p_4,\perp)$. Cette stratégie utilise un compteur κ qui est initialisé à 0, et qui est mis à jour comme suit.

- lorsque l'on vient d'une configuration dont l'état de contrôle est p_4, κ est mis à 0.
- lorsque l'on vient d'une configuration dont l'état de contrôle est p_2, p_1 ou p'_4, κ est incrémenté d'une unité.

– lorsque l'on vient d'une configuration dont l'état de contrôle est p_3, κ est décrémenté d'une unité.

Ainsi, lorsque l'on est dans une configuration d'état de contrôle p'_4, κ est le nombre de a ayant été empilé depuis le dernier passage dans une configuration d'état de contrôle p_2.

Décrivons maintenant comment Eve joue. Depuis une configuration ayant p_3 comme état de contrôle (ce sont les seules où Eve doit jouer), Eve dépile (il sera facile d'établir que le sommet de pile est alors a et que ce coup est donc possible) si et seulement si $\kappa > 0$. Si $\kappa = 0$, Eve joue la transition $push(p_4,b)$

Si Adam reste indéfiniment dans l'état p_4 ou l'état p'_4, la partie est remportée par Eve (la seule couleur apparaissant infiniment souvent est 4). Sinon, on appelle Λ la partie obtenue, et il est facile de voir que $Steps_\Lambda$ contient tous les indices des configurations d'état de contrôle q_2 et aucun des indices des configurations d'état de contrôle q_1. Dès lors, la suite des couleurs sur laquelle est évaluée la condition de parité est décrite par l'expression rationnelle $(4^*23)^\omega$. La partie est donc remportée par Eve, et la stratégie ainsi décrite est bien gagnante pour Eve.

Maintenant, supposons qu'Eve possède une stratégie gagnante φ, dans $\mathbb{G}$ depuis $(p_4,\bot)$, qui utilise une mémoire mem sur un domaine fini M. On va alors construire une partie où Eve respecte φ, mais où elle perd. Pour cela, on considère toutes les parties partielles commençant en $(p_4,\bot)$, et où Adam empile jusqu'à atteindre la configuration $(p_3,a^{M+5}\bot)$. Il y a alors $M+1$ parties différentes (selon le moment où Adam applique la règle $push(p_2,a)$) et il y a donc deux parties Λ et Λ', où mem a la même valeur en $(p_3,a^{M+5}\bot)$. Considérons les extensions de Λ et Λ' jusqu'à atteindre une configuration d'état de contrôle p_4, où Eve respecte φ : toute deux atteignent la même configuration $(p_4,ba^k\bot)$, puisque φ ne peut distinguer Λ de Λ'. Dès lors, pour l'une de ces extensions, la plus petite couleur apparaissant dans l'évaluation de la condition de parité en escalier est impaire. Appelons Λ_1 cette extension. Notons que comme Λ_1 se termine par une configuration ayant b pour sommet de pile, ce dernier ne sera jamais supprimé, et donc la couleur impaire précédente sera dans la suite des couleurs sur laquelle la condition de parité sera évaluée. Maintenant, on peut réitérer le raisonnement en considérant les extensions de Λ_1, où Adam empile $M+5$ symboles a avant d'atteindre une configuration d'état de contrôle p_3. On construit alors une extension Λ_2 de Λ_1 dont la couleur minimale (relative à la condition en escalier) sur la nouvelle partie est impaire. En itérant cette construction, on construit une partie infinie où Eve respecte φ, et qui est remportée par Adam, ce qui conclut la preuve. □

5.6.5 Bornes supérieures et inférieures

On donne enfin la complexité précise des problèmes de décision du gagnant dans les jeux précédents. Les algorithmes proposés réduisent le problème de calcul des ensembles de retours au calcul des positions gagnantes dans un jeu sur un graphe fini de taille exponentielle. Le tableau 5.6.5, récapitule les réductions précédentes (où $n = |Q|$).

Pour la condition de parité, on conclut facilement que l'on peut calculer les ensembles de retours en temps exponentiel. En effet, le jeu de parité $\widetilde{\mathbb{G}}$ a lieu sur

Condition de gain dans $\mathbb{G}$	**Taille de $\widetilde{\mathcal{G}}$**	**condition de gain dans $\widetilde{\mathbb{G}}$**
Parité avec d couleurs	Exponentielle en n et en d	Parité avec d couleurs
Explosion stricte	Exponentielle en n	Büchi
Parité en escalier avec d couleurs	Exponentielle en n	Parité avec d couleurs

TAB. 5.2 – *Récapitulatif des réductions*

un graphe exponentiel, mais le nombre de couleurs intervenant dans la condition de parité est le même que dans le jeu original. Dès lors, en utilisant un algorithme classique pour les jeux de parité sur un graphe fini [46, 80], on calcule les ensembles de retours en temps exponentiel. Un raisonnement similaire permet de conclure qu'il en va de même lorsque l'on considère une condition d'explosion ou une condition de parité en escalier.

Concernant les bornes inférieures, mentionnons un résultat d'I. Walukiewicz sur les jeux d'accessibilité, dont nous donnerons une esquisse de preuve dans le paragraphe 6.4.1

Théorème 5.5 *[83, 84]Décider le gagnant dans un jeu d'accessibilité sur un processus à pile depuis une configuration de pile vide est un problème* DEXPTIME*-dur.*

Dès lors, on a facilement le corollaire suivant.

Corollaire 5.3 *Décider le gagnant dans un jeu sur un processus à pile depuis une configuration de pile vide est un problème* DEXPTIME*-dur pour les conditions de parité, et de parité en escalier.*

Preuve. On réduit les jeux d'accessibilité aux jeux considérés. Pour cela, on modifie la fonction de transition du processus à pile en supprimant toutes les transitions depuis les états finaux. A l'aide de règles skip on ajoute des boucles sur les états finaux. Enfin, on colore par une couleur paire les états finaux et par une couleur impaire les états non finaux. □

Enfin, pour les conditions d'explosion stricte (et donc d'explosion), on peut adapter la preuve d'I. Walukiewicz ou bien encore utiliser les résultats du paragraphe 6.4, et établir le résultat suivant.

Corollaire 5.4 *Décider le gagnant dans un jeu sur un processus à pile depuis une configuration de pile vide est un problème* DEXPTIME*-dur pour les conditions d'explosion stricte et d'explosion.*

En définitive, nous avons prouvé dans ce chapitre le résultat suivant.

Théorème 5.6 *Décider le gagnant dans un jeu sur un processus à pile depuis une configuration de pile vide est un problème* DEXPTIME*-complet pour les conditions de parité, de parité en escalier, d'explosion stricte et d'explosion.*

Chapitre 6

Conditions de gain de complexité borélienne arbitraire finie

Sommaire

Dans le paragraphe 2.3, nous avons introduit la notion de complexité borélienne d'un ensemble de mots et par extension d'une condition de gain. Nous avons aussi noté que les conditions de parité sont des conditions $\mathcal{B}(\Sigma_2)$.

Dans [77], W. Thomas mentionne plusieurs sujets à étudier dont celui-ci :

> Games with winning conditions of Borel level greater than 2. *For finite graphs, even for pushdown graphs, it is reasonable to consider winning conditions which transcend the level of Muller condition, i.e. which are*

> *located on higher levels of the Borel hierarchy, but which still admit algorithmic solutions.*

T. Cachat, J. Duparc et W. Thomas initient dans [21] un premier pas dans cette direction en introduisant la condition de répétition pour les jeux sur des graphes de processus à pile. L'intérêt de leur article est double, puisque la condition de répétition est une condition Σ_3-complète naturelle pour les jeux sur des graphes de processus à pile qui de plus est décidable. L'introduction de [21] se termine par le constat suivant :

> *The main result of this paper may be considered as a first tiny step in a far-reaching proposal of Büchi ([14], p.1171–1172). He considers constructive game presentations by "state-recursions", as they arise in automata theoretic games, and he asks to extend the construction of winning strategies in the form of "recursions" (i.e., algorithmic procedures) from the case of $\mathcal{B}(\Sigma_2)$-games to appropriate games on arbitrary levels of the Borel hierarchy.*

Dans ce chapitre, on fait quelques pas de plus dans la direction indiquée par R. Büchi et W. Thomas, puisqu'on donne deux familles de conditions de gain dont la complexité borélienne est arbitraire mais finie, et qui induisent des jeux décidables sur les graphes finis pour la première et sur les graphes de processus à compteur pour la seconde.

Dans un premier temps, on introduit une nouvelle famille de langages à partir de laquelle on définit les familles de conditions de gain. L'analyse de la complexité borélienne, principalement faite sur la famille de langages, utilise des jeux de Wadge et repose principalement sur un résultat donné par J. Duparc dans [29]. La décidabilité peut être vue comme une généralisation des techniques introduites dans le chapitre 5 pour la résolution des jeux d'explosion qui sont un cas particulier des jeux étudiés dans ce chapitre.

Une large partie de ce chapitre est consacrée à l'étude de la complexité (au sens de la calculabilité) du problème de décision du gagnant dans les jeux considérés. On y montre que ce problème est non élémentaire et qu'il est dur pour la classe des problèmes élémentaires.

On termine enfin par diverses considérations sur les ensembles de positions gagnantes et sur les stratégies gagnantes. Quelques perspectives et problèmes ouverts concluent ce chapitre.

6.1 Deux familles de conditions de gain

6.1.1 Une nouvelle famille de langages

Etant donné un automate à pile déterministe $\mathcal{A}$ et un mot infini u sur l'alphabet d'entrée de $\mathcal{A}$, on dit que $\mathcal{A}$ *explose strictement sa pile en lisant* u, si $\lim(\sigma_i)_{i\geq 0}$ est infinie, où la suite $(\sigma_i)_{i\geq 0}$ est celle des contenus de pile de $\mathcal{A}$ lorsqu'il lit u. On qualifie alors $\lim(\sigma_i)_{i\geq 0}$ de *limite* de la pile de $\mathcal{A}$ lorsqu'il lit u.

Considérons un entier $n \geq 0$ et une collection d'automates à pile déterministes $\mathcal{A}_1, \ldots, \mathcal{A}_n$. Dans le cas particulier où $n = 0$, cette collection est vide. Considérons

enfin un automate à pile déterministe $\mathcal{A}_{n+1}$ équipé d'une condition de parité.

On souhaite faire fonctionner les automates $\mathcal{A}_1, \ldots, \mathcal{A}_{n+1}$ les uns après les autres : $\mathcal{A}_1$ va lire un mot infini donné en entrée. Ensuite, $\mathcal{A}_2$ va lire la limite (dans un sens à préciser) de la pile de $\mathcal{A}_1$ et ainsi de suite jusqu'à ce que $\mathcal{A}_{n+1}$ lise la limite de la pile de $\mathcal{A}_n$, et accepte ou rejette cette dernière à l'aide de sa condition de parité.

On demande donc la consistance suivante entre les alphabets d'entrée et de pile des différents automates. Par la suite, nous supposerons que cette consistance à toujours lieu. On demande donc :

Pour tout $1 \leq i < n$, l'alphabet d'entrée de $\mathcal{A}_{i+1}$ est égal à l'alphabet de pile de $\mathcal{A}_i$.

Appelons A l'alphabet d'entrée de $\mathcal{A}_1$. On associe alors avec $\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}_{n+1}$ un langage de mots infinis sur l'alphabet A, noté $L(\mathcal{A}_1 \rhd \ldots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$, et défini comme suit :

- si $n = 0$, $L(\mathcal{A}_1 \rhd \ldots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}) = L(\mathcal{A}_{n+1})$ est le langage des mots infinis acceptés par $\mathcal{A}_{n+1}$ (au sens de la définition 1.34).
- si $n > 0$, $L(\mathcal{A}_1 \rhd \ldots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$ est l'ensemble des mots infinis u_0 sur l'alphabet A tels que :
  - lorsque $\mathcal{A}_1$ lit u_0, sa pile explose strictement et possède donc une limite u_1.
  - $u_1 \in L(\mathcal{A}_2 \rhd \ldots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$.

De façon équivalente, un mot $u_0 \in A^\omega$ appartient au langage $L(\mathcal{A}_1 \rhd \ldots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$ si et seulement si :

- pour tout $1 \leq i \leq n$, lorsque $\mathcal{A}_i$ lit u_{i-1}, sa pile explose strictement et possède donc une limite u_i.
- enfin, $\mathcal{A}_{n+1}$ accepte u_n.

La figure 6.1 illustre le processus d'acceptation pour $n = 3$.

Enfin, on note $\mathbb{C}_n(A)$ la classe des langages L de mots infinis sur l'alphabet A tel qu'il existe une collection d'automates à pile déterministes $\mathcal{A}_1, \ldots, \mathcal{A}_n$, et un automate à pile déterministe muni d'une condition de parité $\mathcal{A}_{n+1}$, tels que $L = L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$.

En particulier, $\mathbb{C}_0(A)$ est la classe des langages ω-algébriques déterministes sur l'alphabet A introduite dans le paragraphe 1.2.4.

6.1.2 Une famille de conditions de gain externes

Pour tout entier $n \geq 0$, pour toute collection $\mathcal{A}_1, \ldots, \mathcal{A}_n$ d'automates à pile déterministes et pour tout automate à pile déterministe $\mathcal{A}_{n+1}$ muni d'une condition de parité, on définit la condition de gain externe $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}} = L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. Dans la suite, nous considérerons des jeux sur des graphes finis équipés de telles conditions de gain.

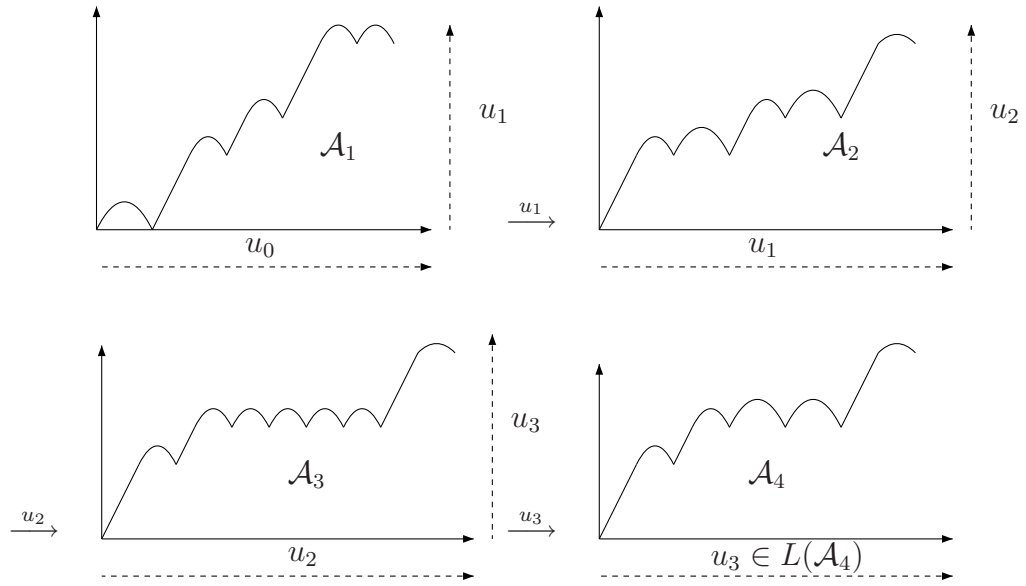


FIG. 6.1 – *Définition du langage* $L(\mathcal{A}_1 \rhd \mathcal{A}_2 \rhd \mathcal{A}_3 \rhd \mathcal{A}_4)$

6.1.3 Une famille de conditions de gain internes pour les jeux sur des graphes de processus à pile

Pour tout entier $n \geq 0$, pour toute collection $\mathcal{A}_1, \ldots, \mathcal{A}_n$ d'automates à pile déterministes, et pour tout automate à pile déterministe $\mathcal{A}_{n+1}$ muni d'une condition de parité, on définit la condition de gain interne $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ pour des jeux sur des graphes de processus à pile par :

$$\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}} = \{\Lambda \mid \Lambda \in \Omega_{ExpSt} \text{ et } \mathrm{StLim}(\Lambda) \in L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})\},$$

où Ω_{ExpSt} désigne la condition d'explosion stricte. Ainsi, une partie est remportée par Eve pour la condition $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ si et seulement si la pile explose strictement, et si sa limite est dans $L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$.

6.2 Complexité borélienne

Dans ce qui suit, on donne la complexité borélienne des langages de $\mathbb{C}_n(A)$. On en déduit celle des conditions de gains $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ et $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$. Les preuves reposent sur la notion de jeu de Wadge introduite dans le paragraphe 2.3.2, et sur une opération ensembliste que l'on présente ci-dessous.

6.2.1 L'opération $X \mapsto X^{\sim}$

Dans [29], J. Duparc introduit trois opérations sur les ensembles boréliens qui sont respectivement homomorphes à la somme pour les ordinaux, la multiplication

par un ordinal dénombrable et l'exponentiation ordinale de base κ, où κ est un ordinal régulier non dénombrable. Ici, nous nous intéressons uniquement à l'opération $X \mapsto X^{\sim}$, qui est donc la contrepartie ensembliste de l'opération d'exponentiation de base κ.

L'opération $X \mapsto X^{\sim}$ est définie pour des ensembles $X \subseteq A^{\infty}$ de mots finis ou infinis sur un alphabet quelconque A. Dès lors, afin de définir une telle opération, on doit transformer les ensembles de mots finis et infinis, en ensembles de mots infinis. La solution adoptée dans [29] consiste à rajouter une nouvelle lettre, assimilée à un symbole blanc, que l'on insère infiniment souvent dans un mot fini pour le transformer alors en un mot infini. Cependant, nous n'allons pas utiliser ce formalisme ici, et nous nous contenterons d'une conséquence des travaux de Duparc qui permet de travailler directement sur des ensembles de mots infinis. C'est pour cela que nous donnons les définitions dans ce cadre là.

Définition 6.1 (def. 22 of [29]) *Soit un alphabet A, soit un ensemble $X \subseteq A^{\omega}$, et soit $\twoheadleftarrow \notin A$ un nouveau symbole appelé* effaceur*, alors $X^{\sim} = \{u \in (A \cup \{\twoheadleftarrow\})^{\omega} \mid u^{\twoheadleftarrow} \in X\}$, où $u^{\twoheadleftarrow}$ est défini par récurrence comme suit :*

- $\varepsilon^{\twoheadleftarrow} = \varepsilon$
- *pour tout mot u fini tel que $|u^{\twoheadleftarrow}| = k$:*
  - $(u \cdot a)^{\twoheadleftarrow} = u^{\twoheadleftarrow} \cdot a$ *si* $a \neq \twoheadleftarrow$.
  - $(u \cdot \twoheadleftarrow)^{\twoheadleftarrow} = u^{\twoheadleftarrow}{\restriction}_{(k-1)}$ *si* $k > 0$ *(on efface la dernière lettre de $u^{\twoheadleftarrow}$).*
  - $(u \cdot \twoheadleftarrow)^{\twoheadleftarrow} = \varepsilon$ *si* $k = 0$ *(il n'y a rien à effacer).*
- *pour u infini, $u^{\twoheadleftarrow} = \lim_{n \in \mathbb{N}}((u{\restriction}_n)^{\twoheadleftarrow})$.*

On a par exemple $ab \twoheadleftarrow\twoheadleftarrow c^{\twoheadleftarrow} = c$, $bb(ab \twoheadleftarrow)^{\omega^{\twoheadleftarrow}} = bba^{\omega}$ et $bb(b \twoheadleftarrow)^{\omega^{\twoheadleftarrow}} = bb$.

L'opération $X \mapsto X^{\sim}$ a une interprétation très naturelle en terme de jeu de Wadge. Effectivement, un joueur en charge de $X^{\sim}$ joue comme s'il était en charge de X, sauf qu'il peut également jouer l'effaceur (ou plusieurs occurrences de ce dernier), et ainsi effacer un nombre arbitraire de symboles déjà écrits.

On peut enfin donner une version itérée de l'opération $X \mapsto X^{\sim}$ définie comme suit.

Définition 6.2 *Soit un ensemble X. Alors on définit $X^{\sim 0} = X$ et pour tout $n \geq 0$, $X^{\sim n+1} = (X^{\sim n})^{\sim}$.*

Par la suite nous utiliserons principalement une conséquence du lemme 31 de [29], énoncée par O. Finkel dans [35] pour des ensembles de mots infinis.

Lemme 6.1 *[35] Soit un alphabet A et soit $X \subseteq A^{\omega}$ un ensemble Π_k-complet pour un $k \geq 2$. Alors, $(X^{\sim n})$ est un ensemble Π_{n+k}-complet, pour tout $n \geq 0$.*

Enfin, on termine par un résultat dû à C. Löding [54].

Lemme 6.2 *Soit un alphabet A et soit un ensemble $X \subseteq A^{\omega}$. Si $X \in \mathcal{B}(\Sigma_n)$ pour un $n \geq 2$ alors $X^{\sim} \in \mathcal{B}(\Sigma_{n+1})$.*

.

Preuve. On commence par considérer le résultat de l'opération $X \mapsto X^\sim$ sur un ensemble ouvert. On prouve que l'on obtient alors un élément de Σ_2. Pour cela, il suffit d'établir le résultat pour un ensemble Σ_1-complet puisque l'opération $X \mapsto X^\sim$ préserve l'ordre de Wadge, c'est-à-dire que si $X \leq_W Y$ alors $X^\sim \leq_W Y^\sim$ [29].

On considère donc le langage $O = (a^*b)A^\omega$ sur l'alphabet $A = \{a,b\}$ qui est Σ_1-complet en tant que complémentaire de l'ensemble Π_1-complet a^ω (voir exemple 2.1).

On a alors :

$$O^\sim = \bigcup_{n\geq 1}((A \cup \{\twoheadleftarrow\})^{n-1}b(A \cup \{\twoheadleftarrow\})^\omega) \cap K_n,$$

où K_n est l'ensemble des mots infinis sur l'alphabet $A \cup \{\twoheadleftarrow\}$ tels que la n-ème lettre de α n'est pas effacée lors du calcul de $\alpha^{\twoheadleftarrow}$. Plus précisément,

$$K_n = \overline{(H_{\geq n}(A \cup \{\twoheadleftarrow\})^\omega)},$$

où $H_{\geq n}$ est l'ensemble des mots finis u tels que la n-ème lettre de u est effacée lors du calcul de $u^{\twoheadleftarrow}$.

Dès lors, K_n est un ensemble Π_1 et donc $O^\sim$ est un ensemble Σ_2.

Maintenant, remarquons que l'opération $X \mapsto X^\sim$ commute avec l'intersection et l'union. Ce n'est en revanche pas le cas en général avec la complémentation, puisque l'on peut avoir $(\overline{X})^\sim \subsetneq \overline{(X^\sim)}$. En effet, $\overline{(X^\sim)}$ contient aussi les mots u tels que $u^{\twoheadleftarrow}$ est fini. On montre alors facilement que $(\overline{X})^\sim = \overline{(X^\sim)} \cap \mathrm{Inf}$, où Inf est l'ensemble des mots $u \in (A \cup \{\twoheadleftarrow\})^\omega$ tels que $u^{\twoheadleftarrow}$ est infini :

$$\mathrm{Inf} = \bigcap_{m\geq 0}\bigcup_{n\geq m} K_n$$

On en conclut donc que Inf est un ensemble Π_3.

Considérons enfin un ensemble X dans $\mathcal{B}(\Sigma_n)$ pour un $n \geq 2$, et appliquons l'opération $X \mapsto X^\sim$ à la formule prouvant son appartenance à $\mathcal{B}(\Sigma_n)$. On pousse alors l'opérateur jusqu'au niveau des ensembles Σ_1 en intersectant avec Inf dès que l'on commute avec une opération de complémentation. La formule ainsi obtenue est comme la formule de départ, sauf que les ouverts ont été transformés en ensembles Σ_2, et que l'on a également ajouté des intersection avec l'ensemble $\mathrm{Inf} \in \Pi_3$. Comme $n \geq 2$, l'intersection avec Inf n'augmente pas la complexité borélienne, et l'on en conclut donc que $X^\sim$ appartient à la classe $\mathcal{B}(\Sigma_{n+1})$. □

6.2.2 Complexité borélienne des langages de $\mathbb{C}_n(A)$

Nous nous intéressons maintenant à la complexité borélienne des langages dans la classe $\mathbb{C}_n(A)$ pour un alphabet A et pour un entier $n \geq 0$.

On a le résultat suivant, qui caractérise précisément la complexité topologique de ces langages.

Théorème 6.1 *Soit un alphabet fini A. Pour tout $n \geq 0$ et pour tout langage L dans $\mathbb{C}_n(A)$, L est dans $\mathcal{B}(\Sigma_{n+2})$. De plus, pour tout $n \geq 0$, il existe un alphabet fini A, et un langage dans $\mathbb{C}_n(A)$ qui est Π_{n+2}-complet.*

Lorsque $n = 0$, le résultat est un classique, puisqu'il ne s'agit ni plus ni moins de dire que tout langage reconnu par une machine déterministe équipée d'une condition de parité est dans $\mathcal{B}(\Sigma_2)$ (la preuve est une simple généralisation de celle montrant que les langages ω-réguliers sont dans $\mathcal{B}(\Sigma_2)$: voir par exemple [70]). Pour ce qui est de la complétude, il suffit dans ce cas de considérer le langage ω-régulier sur l'alphabet $\{a,b\}$ des mots contenant une infinité de a (voir l'exemple 2.2).

Pour le cas général, on procède par récurrence sur n et l'on utilise le lemme suivant.

Lemme 6.3 *Soit un entier $n \geq 1$ et soit une collection $\mathcal{A}_1, \ldots, \mathcal{A}_n$ d'automates déterministes à pile, et soit $\mathcal{A}_{n+1}$ un automate déterministe à pile muni d'une condition de parité. Alors $L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}) \leq_W L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})^\sim$.*

Il existe de plus un automate déterministe à pile $\mathcal{A}_1$ tel que l'inégalité précédente soit une équivalence.

Preuve. Soit $X = L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$ et soit $Y = L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})^\sim$.

On montre que Bob possède une stratégie gagnante dans le jeu de Wadge $G(X,Y)$. En effet, il joue de façon à ce que si v est le mot qu'il a écrit depuis le début de la partie, $v^{\leftarrow\rho}$ est égal au contenu de la pile de $\mathcal{A}_1$ une fois qu'il a lu le mot u écrit par Alice depuis le début de la partie. Il est facile de voir que, grâce à l'effaceur, Bob peut jouer de la sorte. Dès lors, il découle directement des définitions de X et de Y qu'une telle stratégie est gagnante pour Bob. On a donc bien $X \leq_W Y$.

Considérons maintenant le cas particulier où $\mathcal{A}_1 = \langle \{q\}, A, \Gamma, \bot, q, \delta \rangle$ avec :

- $A = \Gamma \cup \{\twoheadleftarrow\}$ où $\twoheadleftarrow \notin \Gamma$.
- pour tout $\gamma \in \Gamma$, $\delta(q,\gamma,a) = push(q,a)$, pour $a \neq \twoheadleftarrow$.
- pour tout $\gamma \in \Gamma$, si $\gamma \neq \bot$, $\delta(q,\gamma,\twoheadleftarrow) = pop(q)$, et $\delta(q,\bot,\twoheadleftarrow) = skip(q)$.

Maintenant, si l'on considère $\twoheadleftarrow$ comme un effaceur, il vient directement que le contenu de la pile de $\mathcal{A}_1$, après avoir lu un mot u, est $u^{\leftarrow\rho}$. De plus, la limite de la pile de $\mathcal{A}_1$ après avoir lu un mot infini u vaut $u^{\leftarrow\rho}$. Par définition du mode d'acceptance, on a $u \in X$ si et seulement si la limite de la pile de $\mathcal{A}_1$ après avoir lu u est infinie et appartient à $L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. On a donc $u \in X$ si et seulement si $u^{\leftarrow\rho} \in L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$, ce qui signifie exactement que $X \equiv_W Y$. □

Le théorème 6.1 découle de la transitivité de l'ordre $\leq_W$, des lemmes 6.1, 6.2 et 6.3 et du cas de base $n = 0$.

6.2.3 Complexité borélienne des conditions de gains $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ et $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$

Nous sommes maintenant prêts pour établir la complexité borélienne des conditions de gains de la forme $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ et $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$.

Pour la version externe, c'est-à-dire pour une condition de la forme $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$, on a le corollaire immédiat du théorème 6.1

Corollaire 6.1 *Pour tout $n \geq 0$ on a :*

- *pour tout collection d'automates déterministes à pile $\mathcal{A}_1, \ldots, \mathcal{A}_n$ et tout automate à pile déterministe muni d'une condition de parité $\mathcal{A}_{n+1}$, $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$ est une condition externe de gain qui est dans $\mathcal{B}(\Sigma_{n+2})$.*

– *il existe une collection d'automates déterministes à pile* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *et un automate à pile déterministe muni d'une condition de parité* $\mathcal{A}_{n+1}$*, tels que* $\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$ *est une condition externe de gain qui est* Π_{n+2}*-complète.*

Pour la version interne, c'est-à-dire pour une condition de la forme $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$, on a le résultat suivant.

Théorème 6.2 *Pour tout* $n \geq 0$ *on a :*

– *pour toute collection d'automates déterministes à pile* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *et tout automate à pile déterministe muni d'une condition de parité* $\mathcal{A}_{n+1}$*,* $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$ *est une condition interne de gain qui est dans* $\mathcal{B}(\Sigma_{n+3})$*.*

– *il existe une collection d'automates déterministes à pile* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *et un automate à pile déterministe muni d'une condition de parité* $\mathcal{A}_{n+1}$*, tels que* $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$ *est une condition externe de gain qui est* Π_{n+3}*-complète.*

Le théorème 6.2 est une conséquence directe du théorème 6.1 et du lemme suivant.

Lemme 6.4 *Considérons un jeu sur un graphe de processus à pile* $\mathcal{G}$ *équipé d'une condition de gain de la forme* $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$*. Alors*

$$\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}} \equiv_W L(\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1})^{\sim}$$

Preuve. Soit $X = \Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$ et soit $Y = L(\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1})^{\sim}$

Considérons le jeu de Wadge $W(X,Y)$. Bob possède une stratégie gagnante consistant à faire en sorte que si u est le mot écrit depuis le début de la partie par Bob, $u^{\leftarrow \wp}$ est égal au contenu de la pile dans la dernière configuration écrite par Alice (l'alphabet sur lequel cette dernière joue est l'ensemble des sommets de $\mathcal{G}$).

Réciproquement, considérons le jeu de Wadge $W(Y,X)$. Une stratégie gagnante pour Bob est d'écrire une configuration de contenu de pile égale à $u^{\leftarrow \wp}$, où u est le mot écrit jusque là par Alice. □

6.3 Décidabilité

Dans ce qui suit, nous expliquons comment décider le gagnant dans un jeu sur un graphe fini équipé d'une condition de gain de la forme $\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$ et dans un jeu sur un graphe de processus à pile et équipé d'une condition de gain de la forme $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$. Plus précisément, on établit le résultat suivant.

Théorème 6.3 *Pour toute collection d'automates déterministes à pile* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *et tout automate à pile déterministe muni d'une condition de parité* $\mathcal{A}_{n+1}$*, on a les deux points suivants :*

– *soit* $\mathcal{G}$ *un graphe de jeu fini. On peut alors décider, pour tout sommet* v *dans* $\mathcal{G}$*, si Eve possède une stratégie gagnante depuis* v *dans le jeu* $\mathbb{G} = (\mathcal{G}, \Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}})$*.*

– *soit* $\mathcal{G}$ *un graphe de processus à pile. On peut alors décider, pour toute configuration de pile vide* $(q, \bot)$ *dans* $\mathcal{G}$*, si Eve possède une stratégie gagnante depuis* $(q, \bot)$ *dans le jeu* $\mathbb{G} = (\mathcal{G}, \Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}})$*.*

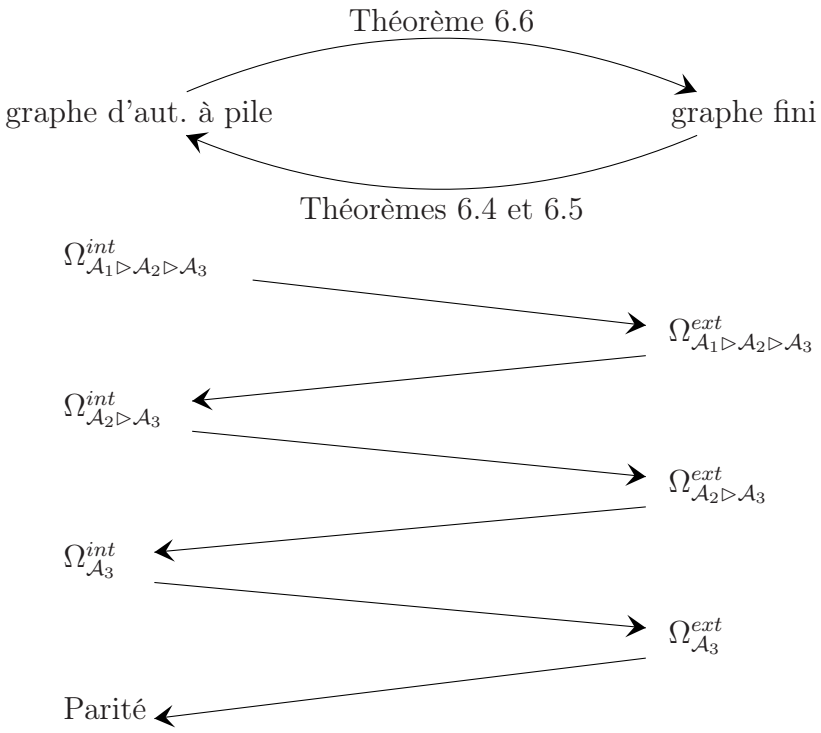


FIG. 6.2 – *Preuve du théorème 6.3 lorsque* $n = 2$

La technique pour résoudre de tels jeux est la suivante : on transforme un jeu sur un graphe fini en un jeu sur un graphe de processus à pile avec une condition de gain plus simple, c'est-à-dire faisant intervenir un automate de moins (voir le théorème 6.4). Dans le cas particulier où la condition de départ est de la forme $\Omega^{ext}_{\mathcal{A}_1}$, le jeu obtenu est un jeu sur un graphe de processus à pile muni d'une condition de parité (voir le théorème 6.5), et l'on sait donc le résoudre (voir le chapitre 5). Enfin, on donne une transformation d'un jeu sur un graphe de processus à pile en un jeu sur un graphe fini avec une condition de gain faisant intervenir le même nombre d'automates à pile (voir le théorème 6.6). La figure 6.2 illustre ce raisonnement lorsque $n = 2$.

Les deux prochaines sous-sections sont consacrées à la description et à la preuve de ces transformations.

6.3.1 D'un graphe fini à un graphe de processus à pile

On considère dans toute ce paragraphe un graphe de jeu fini, noté $\widetilde{\mathcal{G}} = ((V,E),V_{\mathbf{E}},V_{\mathbf{A}})$, dans lequel les arcs sont étiquetés par des lettres prises dans un alphabet fini A ou par le mot vide ε.

On se donne également un entier $n \geq 0$ ainsi qu'une collection $\mathcal{A}_1, \ldots, \mathcal{A}_n$ d'automates à pile déterministes, et l'on considère un automate à pile déterministe muni d'une condition de parité $\mathcal{A}_{n+1}$. L'automate $\mathcal{A}_1$ a pour alphabet d'entrée A. Enfin, on considère une position initiale $v_{in} \in V$ dans $\widetilde{\mathcal{G}}$.

On souhaite décider si Eve possède une stratégie gagnante depuis v_{in} dans le jeu $\widetilde{\mathbb{G}} = (\widetilde{\mathcal{G}},\Omega^{ext}_{\mathcal{A}_1\triangleright\cdots\triangleright\mathcal{A}_n\triangleright\mathcal{A}_{n+1}})$. Pour cela, on construit un jeu sur un graphe de processus à pile, équipé de la condition $\Omega^{int}_{\mathcal{A}_2\triangleright\cdots\triangleright\mathcal{A}_n\triangleright\mathcal{A}_{n+1}}$ si $n \geq 1$, et équipé d'une condition de parité si $n = 0$. Enfin, on montre (théorèmes 6.4 et 6.5) qu'il existe une position dans le nouveau jeu qui est gagnante pour Eve si et seulement si v_{in} est gagnante dans $\widetilde{\mathbb{G}} = (\widetilde{\mathcal{G}},\Omega^{ext}_{\mathcal{A}_1\triangleright\cdots\triangleright\mathcal{A}_n\triangleright\mathcal{A}_{n+1}})$.

L'idée pour transformer $\widetilde{\mathbb{G}}$ en un jeu *équivalent* sur un graphe de processus à pile est proche de celles utilisées dans le chapitre 4 (en particulier pour les conditions ω-régulières sur les états) : on code le calcul de $\mathcal{A}_1$ sur une partie dans $\widetilde{\mathcal{G}}$ dans le graphe produit de $\widetilde{\mathcal{G}}$ avec $\mathcal{A}_1$.

Plus précisément, si l'on note $\mathcal{A}_1 = \langle Q,\Gamma,A,\bot,q_{in},\delta\rangle$, on définit le processus à pile $\mathcal{P} = \langle Q \times V,\Gamma,\bot,\Delta\rangle$ où :

- $skip((q',v')) \in \Delta((q,v),\gamma)$ si et seulement s'il existe une étiquette $a \in A$ telle que $(v,a,v') \in E$ et $\delta(q,\gamma,a) = skip(q')$, ou si $(v,\varepsilon,v') \in E$ et $q = q'$.
- $pop((q',v')) \in \Delta((q,v),\gamma)$ si et seulement s'il existe une étiquette $a \in A$ telle que $(v,a,v') \in E$ et $\delta(q,\gamma,a) = pop(q')$.
- $push((q',v'),\gamma') \in \Delta((q,v),\gamma)$ si et seulement s'il existe une étiquette $a \in A$ telle que $(v,a,v') \in E$ et $\delta(q,\gamma,a) = push(q',\gamma')$.

On considère alors la partition de $Q \times V$ induite par la partition de V, $Q \times V = Q \times V_{\mathbf{E}} \cup Q \times V_{\mathbf{A}}$, et l'on appelle $\mathcal{G}$ le graphe de processus à pile induit par $\mathcal{P}$ et la partition précédente. Intuitivement, $\mathcal{G}$ code un calcul à la volée de $\mathcal{A}_1$ sur une partie dans $\widetilde{\mathcal{G}}$. Concernant l'équivalence entre les jeux, on a le résultat suivant dans le cas où $n \geq 1$

Théorème 6.4 *Si $n \geq 1$, pour tout sommet $v_{in} \in V$, Eve possède une stratégie gagnante depuis v_{in} dans $\widetilde{\mathbb{G}} = (\widetilde{\mathcal{G}},\Omega^{ext}_{\mathcal{A}_1\triangleright\cdots\triangleright\mathcal{A}_n\triangleright\mathcal{A}_{n+1}})$ si et seulement si elle possède une stratégie gagnante depuis $((q_{in},v_{in}),\bot)$ dans $\mathbb{G} = (\mathcal{G},\Omega^{int}_{\mathcal{A}_2\triangleright\cdots\triangleright\mathcal{A}_n\triangleright\mathcal{A}_{n+1}})$.*

Preuve. Supposons qu'Eve possède une stratégie gagnante φ depuis v_{in} dans $\widetilde{\mathbb{G}}$. On définit alors une stratégie Φ dans $\mathbb{G}$ depuis $((q_{in},v_{in}),\bot)$. Comme le graphe $\mathcal{G}$ code des calculs à la volée de $\mathcal{A}_1$ sur des parties dans $\widetilde{\mathcal{G}}$, la stratégie Φ va reconstruire une partie dans $\widetilde{\mathcal{G}}$ (à l'aide d'une fonction notée τ), et va utiliser la valeur de φ sur cette nouvelle partie pour choisir quel coup jouer. La stratégie Φ utilise également δ pour mettre à jour la pile et la première composante de l'état de contrôle. Une telle construction est à rapprocher de la notion d'enrichissement déterministe donnée dans le paragraphe 4.3.2.

Pour simplifier les notations, une partie $\lambda = (v_0,e_1e_2e_3\cdots)$ dans $\widetilde{\mathcal{G}}$ sera représentée par le mot $v_0a_1v_1a_2v_2\cdots$ où $e_i = (v_i,a_i,v_{i+1})$, pour tout $1 \leq i$.

Soit Λ une partie partielle dans $\mathbb{G}$ commençant en $((q_{in},v_{in}),\bot)$, et soit τ l'application des parties jouées sur $\mathcal{G}$ dans les parties jouées sur $\widetilde{\mathcal{G}}$, définie récursivement comme suit :

- si $\Lambda = ((q_{in},v_{in}),\bot)$, alors $\tau(\Lambda) = v_{in}$.
- si $\Lambda = \Lambda' \cdot ((q',v'),\sigma')$, appelons $((q,v),\sigma)$ le dernier sommet de Λ'. On pose alors $\tau(\Lambda) = \tau(\Lambda') \cdot a \cdot v'$ pour un certain $a \in A \cup \{\varepsilon\}$ tel que $(v,a,v') \in E$ et tel que (q',σ') soit le successeur de (q,σ) par a dans $\mathcal{A}_1$. Notons que, par définition de $\mathcal{P}$, $\tau(\Lambda)$ est toujours défini (mais pas forcément de façon unique).

Pour toute partie partielle Λ se terminant dans une configuration $((q,v),\sigma)$, on pose $\Phi(\Lambda) = ((q',v'),\sigma')$ où $(v,a,v') = \varphi(\tau(\Lambda))$ pour un $a \in A \cup \{\varepsilon\}$ et où (q',σ') désigne le successeur par a de (q,σ) dans $\mathcal{A}_1$.

A présent, il est facile de voir que toute partie partielle Λ dans $\mathcal{G}$ qui commence en $((q_{in},v_{in}),\bot)$, et dans laquelle Eve respecte Φ, est telle que :

1. $\tau(\Lambda)$ est une partie partielle dans $\widetilde{\mathcal{G}}$ commençant en v_{in} et dans laquelle Eve respecte φ.
2. le calcul de $\mathcal{A}_1$ sur $\mathrm{Lab}(\tau(\Lambda))$ se termine dans la configuration (q,σ), où le dernier sommet de Λ est de la forme $((q,v),\sigma)$ pour un certain $v \in V$.

On conclut qu'Eve remporte toute partie Λ dans $\mathcal{G}$ où elle respecte Φ. En effet, si la partie est finie, le premier point nous permet de conclure en remarquant que le joueur qui ne peut bouger est Adam. Dans le cas où Λ est infinie, $\tau(\Lambda)$ est remportée par Eve. Lorsque $\mathcal{A}_1$ lit $\mathrm{Lab}(\tau(\Lambda))$, sa pile explose strictement, et sa limite est dans $L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. Dès lors, la pile dans Λ explose strictement, et sa limite est dans $L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. On a donc bien que $\Lambda \in \Omega^{int}_{\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$.

Réciproquement, supposons qu'Eve possède une stratégie gagnante Φ depuis $((q_{in},v_{in}),\bot)$ dans le jeu $\mathbb{G}$. A toute partie partielle λ dans $\widetilde{\mathcal{G}}$ commençant en v_{in}, on associe de façon récursive un partie dans $\mathcal{G}$, $\pi(\lambda)$ comme suit :

- si $\lambda = v_{in}$, alors $\pi(\lambda) = ((q_{in},v_{in}),\bot)$.
- si $\lambda = \lambda' \cdot av'$, notons $((q,v),\sigma)$ le dernier sommet de $\pi(\lambda')$. On pose alors $\pi(\lambda) = \pi(\lambda') \cdot ((q',v'),\sigma')$ où (q',σ') est le successeur par a de (q,σ) dans $\mathcal{A}_1$.

Pour tout partie partielle λ se terminant dans un sommet v, on appelle $((q,v),\sigma)$ la dernière configuration de $\pi(\lambda)$, et l'on note $((q',v'),\sigma') = \Phi(\pi(\lambda))$. Il existe alors toujours un $a \in A \cup \{\varepsilon\}$, qui n'est pas forcément unique, tel que (q',σ') est le successeur dans $\mathcal{A}_1$ par a de (q,σ). Enfin, on pose $\varphi(\lambda) = (v,a,v')$.

On voit alors facilement que pour toute partie partielle λ dans $\widetilde{\mathcal{G}}$ débutant en v_{in}, et dans laquelle Eve respecte sa stratégie φ, on a les points suivant :

1. la partie $\pi(\Lambda)$ débute en $((q_{in},v_{in}),\bot)$ et Eve y respecte Φ.
2. le calcul de $\mathcal{A}_1$ sur $\mathrm{Lab}(\lambda)$ se termine dans une configuration (q,σ), où le dernier de $\pi(\lambda)$ est de la forme $((q,v),\sigma)$, pour un certain un certain sommet $v \in V$.

On conclut alors facilement qu'Eve remporte toute partie λ dans $\widetilde{\mathcal{G}}$ où elle respecte φ. En effet, si la partie est finie, le premier point nous permet de conclure que le joueur qui est bloqué est Adam. Dans le cas où la partie λ est infinie, $\pi(\lambda)$ est remportée par Eve. Dans λ, la pile explose strictement et possède de plus une limite dans $L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. La pile de $\mathcal{A}_1$, lorsqu'il lit $\mathrm{Lab}(\lambda)$, explose donc strictement et sa limite est dans $L(\mathcal{A}_2 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. Dès lors, la partie λ est remportée par Eve pour la condition de gain externe $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$. □

Pour le cas où $n = 0$, l'automate $\mathcal{A}_1$ est équipé d'une fonction de coloriage $\rho : Q \to C$ et d'une condition de parité. On étend alors ρ en une application $\rho : Q \times V \to C$ en posant $\rho(q,v) = \rho(q)$. Enfin, on appelle $\mathbb{G}$ le jeu de parité sur le graphe de processus à pile $\mathcal{G}$ muni du coloriage induit par ρ. Les mêmes techniques que celles utilisées pour prouver le théorème 6.4 permettent d'obtenir le résultat suivant

Théorème 6.5 *Pour tout sommet $v_{in} \in V$, Eve possède une stratégie gagnante depuis v_{in} dans $\widetilde{\mathbb{G}} = (\widetilde{\mathcal{G}},\Omega^{ext}_{\mathcal{A}_1})$ si et seulement si elle possède une stratégie gagnante depuis $((q_{in},v_{in}),\bot)$ dans le jeu de parité $\mathbb{G}$.*

6.3.2 D'un graphe de processus à pile à un graphe fini

On considère maintenant un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ ainsi qu'une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de Q. On appelle $\mathcal{G}$ le graphe de jeu associé. On se donne également un entier $n \geq 0$, une collection $\mathcal{A}_1,\ldots,\mathcal{A}_n$ d'automates à pile déterministes et un automate à pile déterministe $\mathcal{A}_{n+1}$ équipé d'une condition de parité. De plus, $\mathcal{A}_1$ possède Γ comme alphabet d'entrée. On considère le jeu $\mathbb{G} = (\mathcal{G},\Omega^{int}_{\mathcal{A}_1\rhd\cdots\rhd\mathcal{A}_n\rhd\mathcal{A}_{n+1}})$ ainsi qu'une configuration initiale de pile vide $(p_{in},\bot)$.

On souhaite maintenant construire un graphe de jeu fini $\widetilde{\mathcal{G}}$ et munir ce dernier de la condition externe $\Omega^{ext}_{\mathcal{A}_1\rhd\cdots\rhd\mathcal{A}_n\rhd\mathcal{A}_{n+1}}$ afin d'avoir l'équivalence suivante : Eve possède une stratégie gagnante depuis $(p_{in},\bot)$ dans $\mathbb{G}$ si et seulement si elle possède une stratégie gagnante dans le jeu $\widetilde{\mathbb{G}} = (\widetilde{\mathcal{G}},\Omega^{ext}_{\mathcal{A}_1\rhd\cdots\rhd\mathcal{A}_n\rhd\mathcal{A}_{n+1}})$ depuis une position particulière.

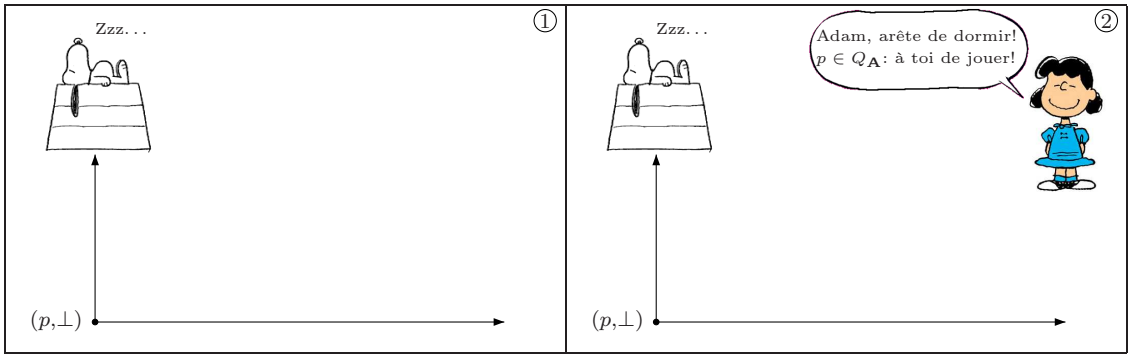
Comme pour les conditions étudiées dans le chapitre 5, la notion de M/B factorisation joue un rôle prépondérant. On a l'analogue de la proposition 5.1 pour les conditions de la forme $\Omega^{int}_{\mathcal{A}_1\rhd\cdots\rhd\mathcal{A}_n\rhd\mathcal{A}_{n+1}}$

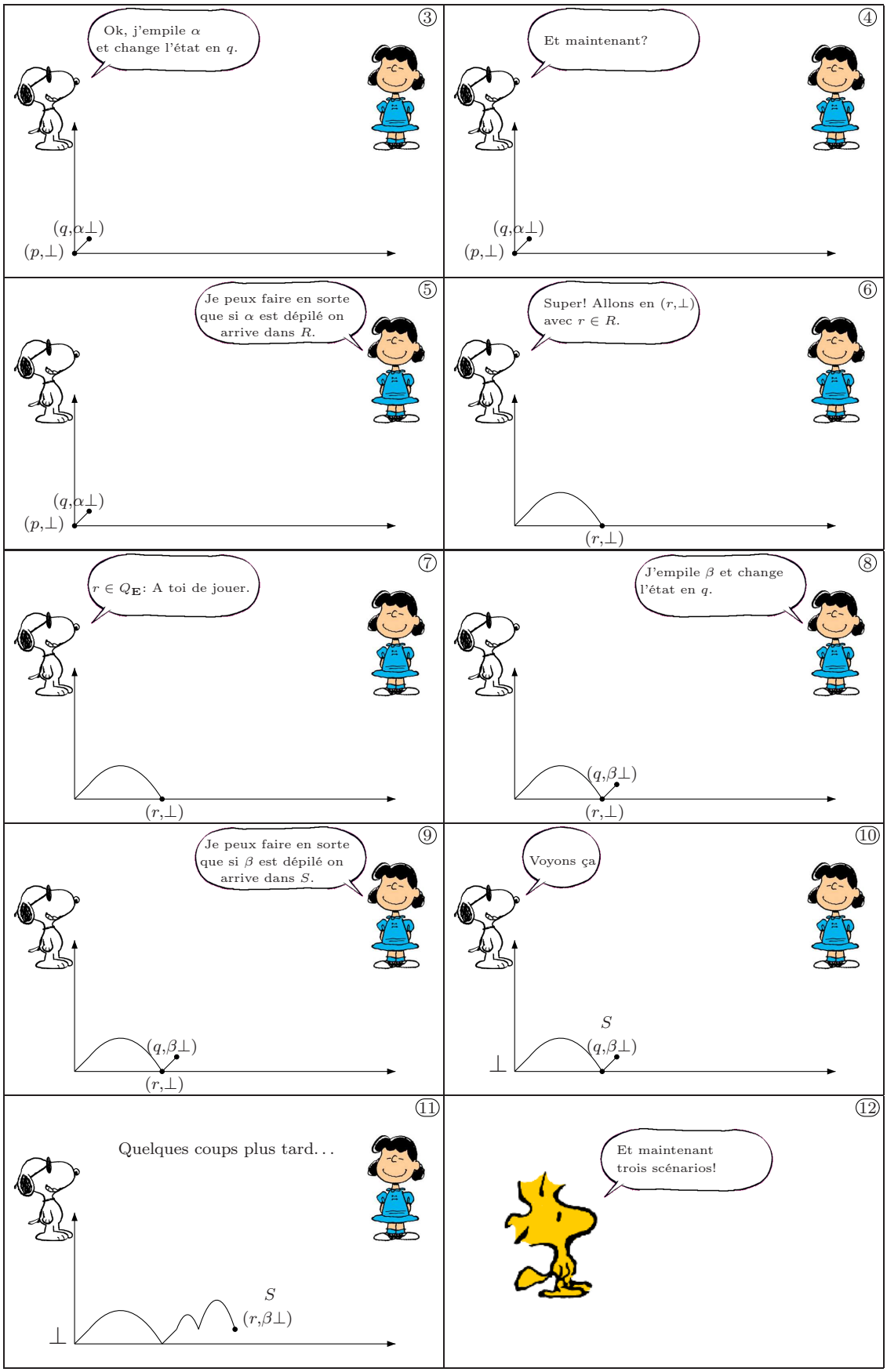
Proposition 6.1 *Une partie Λ dans $\mathbb{G}$ est remportée par Eve si et seulement si Λ est infinie et :*

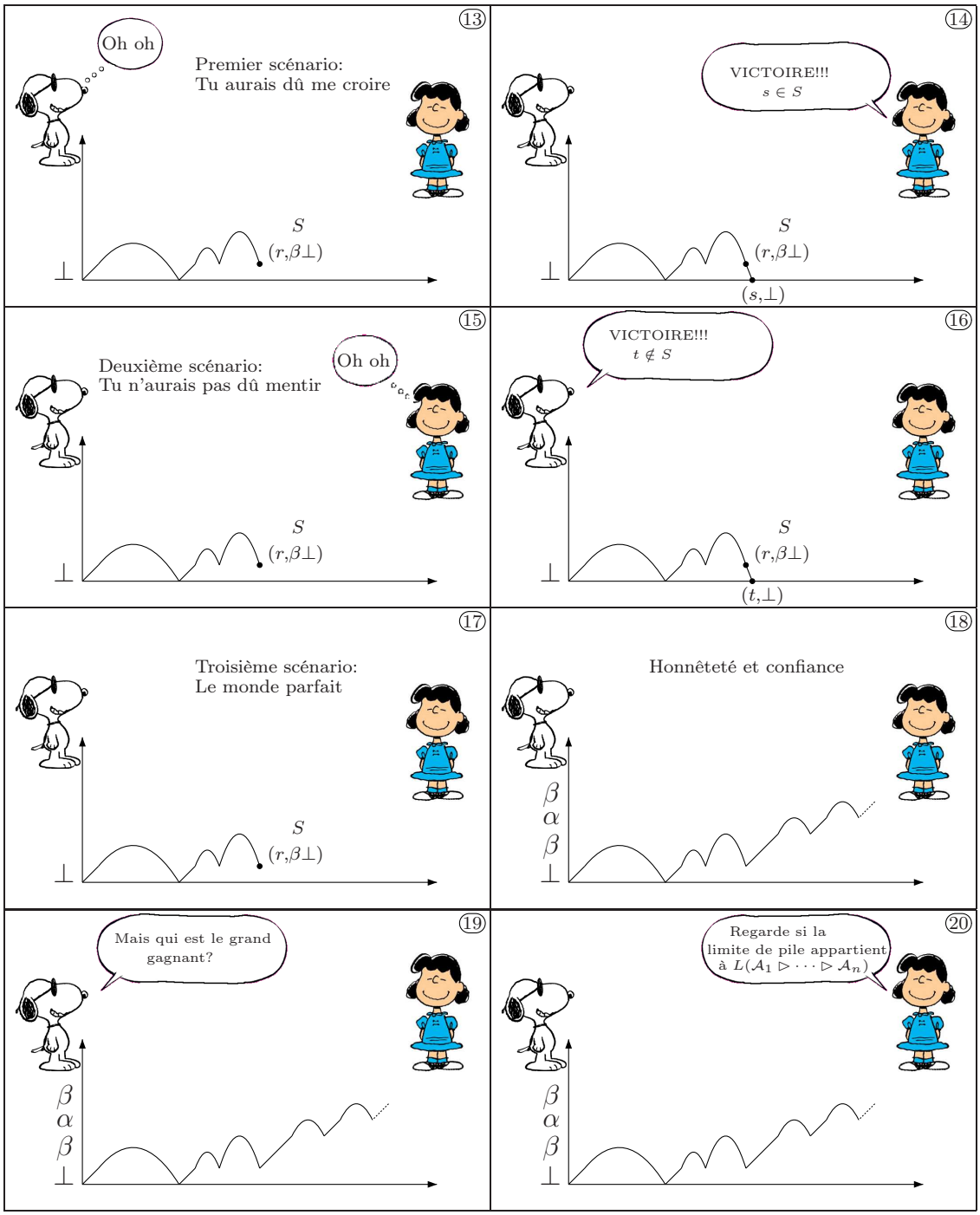
- *il existe une infinité de marches dans la M/B factorisation de Λ.*
- *le mot infini formé par les sommets de pile dans les premières configuration de chaque marche de la M/B factorisation de Λ est dans $L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_{n-1} \rhd \mathcal{A}_n)$.*

Preuve. La preuve est immédiate : le premier point exprime que la pile explose strictement, tandis que le second traduit la condition que la limite de pile est dans $L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_{n-1} \rhd \mathcal{A}_n)$. □

On peut alors imaginer à nouveau un jeu informel dans lequel Eve et Adam se mettent directement d'accord sur la M/B factorisation et ses propriétés. Comme dans le chapitre 5, on illustre cette réduction par une bande dessinée avant de la formaliser dans un jeu sur un graphe fini.




③
Ok, j'empile α et change l'état en q.
$(q,\alpha\perp)$
$(p,\perp)$
④
Et maintenant?
$(q,\alpha\perp)$
$(p,\perp)$
⑤
Je peux faire en sorte que si α est dépilé on arrive dans R.
$(q,\alpha\perp)$
$(p,\perp)$
⑥
Super! Allons en $(r,\perp)$ avec $r \in R$.
$(r,\perp)$
⑦
$r \in Q_{\mathbf{E}}$: A toi de jouer.
$(r,\perp)$
⑧
J'empile β et change l'état en q.
$(q,\beta\perp)$
$(r,\perp)$
⑨
Je peux faire en sorte que si β est dépilé on arrive dans S.
$(q,\beta\perp)$
$(r,\perp)$
⑩
Voyons ça
S
$(q,\beta\perp)$
$\perp$
⑪
Quelques coups plus tard...
S
$(r,\beta\perp)$
$\perp$
⑫
Et maintenant trois scénarios!



TAB. 6.1 – Simulation d'une partie

Afin de simuler une partie dans le jeu $\mathbb{G}$ depuis une configuration $(p,\bot)$, Eve et Adam procèdent de la façon suivante. Le joueur à qui appartient la configuration $(p,\bot)$ peut choisir, a priori, d'empiler, de dépiler, ou de ne pas modifier la pile (vignettes 1 à 2).

Dans le cas où il ne modifie pas la pile, c'est-à-dire qu'une règle de la forme $skip(q) \in \Delta(p,\bot)$ est appliquée, le facteur correspondant n'est autre que la bosse triviale $(p,\bot)(q,\bot)$.

Dans le cas où il décide d'empiler (vignette 3), c'est-à-dire d'appliquer une règle de la forme $push(q,\alpha) \in \Delta(p,\bot)$, on souhaite décider si $(p,\bot)(q,\alpha\bot)$ sera une marche ou le préfixe d'une bosse. Bien sûr, cela dépend de la manière dont Eve et Adam vont décider de jouer. Eve donne alors à Adam des informations sur la stratégie qu'elle va adopter, à savoir un sous-ensemble $R \subseteq Q$ d'états pouvant être atteints si le α est dépilé (vignette 5). Adam peut alors prolonger la partie en une bosse (vignette 6), et donc continuer la partie depuis une position, de son choix, $(r,\bot)$ avec $r \in R$ (vignette 7 à 9), ou continuer avec la nouvelle lettre comme sommet de pile (vignettes 10 avec β comme sommet de pile et S comme ensemble). Dans ce dernier cas, la partie se prolonge depuis $(q,\beta\bot)$, si β désigne le nouveau sommet de pile. On sait également que l'ancien sommet de pile (ici $\bot$) sera dans la limite de pile, et l'on écrit donc cet ancien sommet à la suite (ici au début) de la description de la limite de pile. Par ailleurs, si β est un jour dépilé, Eve gagne (vignettes 13 et 14) si l'état de contrôle est dans S (S est l'ensemble annoncé par Eve précédemment) et sinon c'est Adam (vignettes 15 et 16). Ainsi, la nature de ce second choix est double du point de vue d'Adam : il peut lui permettre de prouver que l'ensemble S donné par Eve est trop petit (il existe un état qui n'est pas dans S et qui est obtenu en dépilant β) ou simplement de simuler une partie où β ne sera jamais dépilé.

Quel que soit le choix d'Adam, les deux joueurs se sont mis d'accord sur le premier facteur de la M/B factorisation. De plus, dans le cas où il s'agit d'une marche, ils connaissent une nouvelle lettre de la limite. Dès lors, au cours d'une partie infinie (vignettes 17 à 20), ils peuvent décider qui est le gagnant, puisque la condition sur la M/B factorisation demande que la limite soit infinie (infinité de marche) et appartienne à $L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. Si l'action d'écrire une lettre de la limite est vue comme le passage dans un arc étiqueté par cette lettre, la nouvelle condition de gain est alors $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$.

Dans le jeu précédent, les seules informations dont les joueurs aient à se rappeler sont le sommet de pile de la configuration actuellement simulée, l'état de contrôle ainsi que l'ensemble des états atteignables en dépilant, qu'Eve a annoncés. On considère donc le graphe de jeu $\widetilde{\mathcal{G}}$ représenté dans la figure 6.3 et dont voici la description :

- les sommets principaux de $\widetilde{\mathcal{G}}$ sont ceux de la forme (p,γ,R), où $p \in Q$, $\gamma \in \Gamma$ et $R \subseteq Q$. Un sommet (p,γ,R) représente une partie partielle Λ dans $\mathbb{G}$ telle que :
  - le dernier sommet de Λ est de la forme $(p,\gamma\sigma)$ pour un certain $\sigma \in \Gamma^*$.
  - Eve prétend pouvoir jouer depuis Λ de telle sorte que si γ est un jour dépilé, l'état de contrôle atteint après avoir dépilé γ est dans R.

  Un sommet de la forme (p,γ,R) appartient à Eve si et seulement si $p \in Q_{\mathbf{E}}$.
- les états *tt* et *ff* sont là pour s'assurer que les ensembles R dans les sommets principaux sont corrects. Le sommet *tt* appartient à Adam tandis que *ff* à Eve. Ces sommets étant des culs-de-sac, une partie qui arrive qui arrive dans *tt* sera remportée par Eve, tandis qu'une partie arrivant dans *ff* sera reportée par Adam.

  Il y a une transition d'un sommet (p,γ,R) vers *tt* si et seulement s'il existe une règle de la forme $pop(r) \in \Delta(p,\gamma)$ telle que $r \in R$ (ce qui traduit le fait que R est correct par rapport à cette règle). Symétriquement, il y a une transition

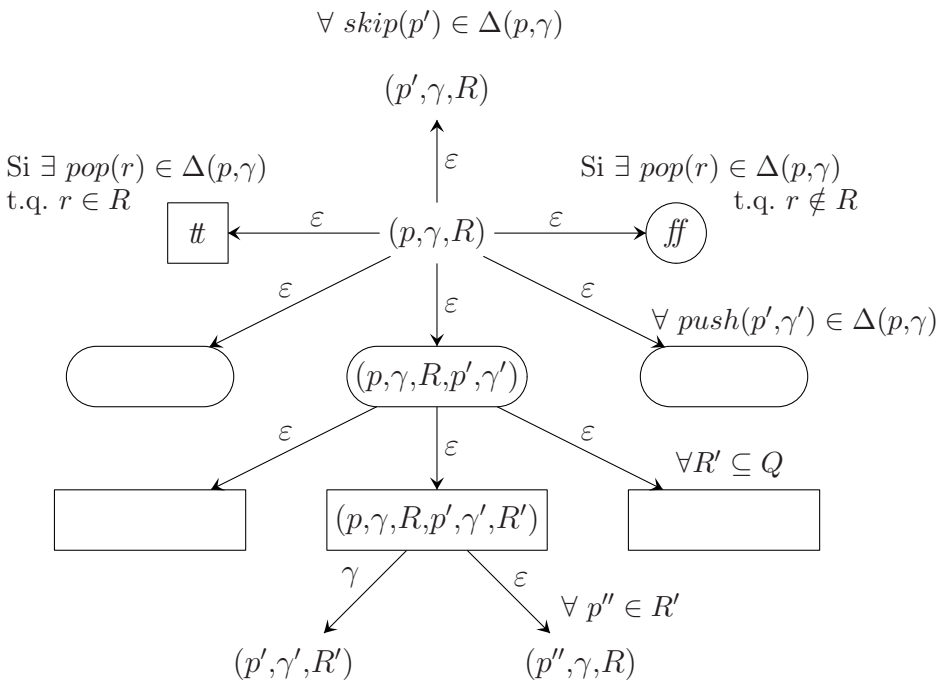


FIG. 6.3 – *Structure locale de* $\widetilde{\mathcal{G}}$ *pour les conditions* $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$.

d'un sommet (p,γ,R) vers *ff* si et seulement s'il existe une règle de la forme $pop(r) \in \Delta(p,\gamma)$ telle que $r \notin R$ (ce qui traduit le fait que R n'est pas correct par rapport à cette règle).

– afin de simuler l'application d'une transition $skip(p') \in \Delta(p,\gamma)$, le joueur à qui appartient (p,γ,R) va en (p',γ,R).
– afin de simuler l'application d'une transition $push(p',\gamma') \in \Delta(p,\gamma)$, le joueur à qui appartient (p,γ,R) va en (p,γ,R,p',γ'). Ce dernier appartient à Eve et celle-ci doit alors choisir un sous-ensemble sous-ensemble R' de Q décrivant les états pouvant être est un jour dépilé. Eve va pour cela dans le sommet le sommet $(p,\gamma,R,p',\gamma',R')$.

  Ce dernier sommet appartient à Adam. Depuis $(p,\gamma,R,p',\gamma',R')$, Adam choisit de simuler une bosse ou une marche. Dans le premier cas, il va dans un sommet (p'',γ,R), pour un état $p'' \in R'$. Dans le second cas, Adam va dans le sommet (p',γ',R').

Le graphe de jeu $\mathcal{G}$ est Γ-étiqueté sur les arc de la façon suivante : tous les arcs simulant une marche, c'est-à-dire allant d'un sommet $(p,\gamma,R,p',\gamma',R')$ à un sommet (p',γ',R'), sont étiquetés par γ. Les autres arcs n'ont pas d'étiquetage, c'est-à-dire qu'ils sont étiquetés par le mot vide ε. Il est important de noter qu'il peut donc y avoir deux arcs allant d'un sommet (p,γ,R,p,γ,R) au sommet (p,γ,R), l'un étiqueté par γ et l'autre par ε, dans le cas particulier où $p \in R$: le premier correspond à la simulation d'une marche tandis que le second correspond à la simulation d'une bosse.

Dans la suite, pour simplifier les notations, une partie $\lambda = (v_0,e_1e_2e_3\cdots)$ dans $\widetilde{\mathcal{G}}$ sera représentée par le mot $v_0a_1v_1a_2v_2\cdots$ où $e_i = (v_i,a_i,v_{i+1})$, pour tout $1 \leq i$.

On a alors le résultat suivant, qui montre la correspondance entre les jeux $\mathbb{G}$ et $\widetilde{\mathbb{G}} = (\widetilde{\mathcal{G}},\Omega^{ext}_{\mathcal{A}_1\rhd\cdots\rhd\mathcal{A}_n\rhd\mathcal{A}_{n+1}})$

Théorème 6.6 *Eve possède une stratégie gagnante depuis une position $(p_{in},\bot)$ dans $\mathbb{G}$ si et seulement si elle possède une stratégie gagnante depuis $(p_{in},\bot,\emptyset)$ dans le jeu $\widetilde{\mathbb{G}}$.*

Comme pour les preuves du chapitre 5, on introduit les concepts de round et de M/B factorisation dans $\widetilde{\mathbb{G}}$. Les définitions sont calquées sur celles du paragraphe 5.2.1.

Une partie dans $\widetilde{\mathbb{G}}$ visite *régulièrement*, au plus tous les trois coups, des sommets de la forme (p,γ,R). On qualifie de *round*, la séquence de sommets entre deux passages dans de telles positions. Un round peut alors être de plusieurs types :

- il est de la forme $(p,\gamma,R)(p',\gamma,R)$ et correspond donc à la simulation d'une règle de type skip. On parle alors de *bosse (triviale)*.
- il est de la forme $(p,\gamma,R)(p,\gamma,R,p',\gamma')(p,\gamma,R,p',\gamma',R')(p'',\gamma,R)$ et correspond donc à la simulation d'une règle empilant un γ' suivie d'une série de coups se terminant par le dépilement du γ'. On le qualifie alors de *bosse*.
- il est de la forme $(p,\gamma,R)(p,\gamma,R,p',\gamma')(p,\gamma,R,p',\gamma',R')(p',\gamma',R')$ et correspond donc à la simulation d'une règle d'empilement d'un γ' qui ne sera pas dépilé. On le qualifie alors de *marche*.

Pour une partie $\lambda = v_0v_1v_2\cdots$ dans $\widetilde{\mathbb{G}}$, on considère le sous-ensemble d'indices correspondant à des sommets de la forme (p,γ,R). Plus précisément :

$$\mathit{Steps}_\lambda = \{n \in \mathbb{N} \mid v_n = (p,\gamma,R),\ p \in Q,\ \gamma \in \Gamma,\ R \subseteq Q\}$$

L'ensemble Steps_λ induit une factorisation naturelle d'un partie λ en bosses et en marches.

Définition 6.3 (M/B factorisation) *Etant donnée une partie, partielle ou non, $\lambda = v_0v_1v_2\cdots$, on appelle* Marche/Bosse factorisation *de λ ou plus simplement* M/B factorisation*, la suite, finie or infinie, $(\lambda_i)_{i\geq 0}$ de rounds de λ définis de la façon suivante. Soit $\mathit{Steps}_\lambda = \{n_0 < n_1 < n_2 < \cdots\}$, on pose alors, pour tout $0 \leq i < |\mathit{Steps}_\lambda|$, $\lambda_i = v_{n_i}\cdots v_{n_{i+1}}$.*

Ainsi, pour tout $i \geq 0$, le premier sommet de λ_{i+1} est le dernier sommet de λ_i. De plus $\lambda = \lambda_1 \odot \lambda_2 \odot \lambda_3 \odot \cdots$, où $\lambda_i \odot \lambda_{i+1}$ désigne la concaténation de λ_i et de λ_{i+1} privé de son premier sommet.

Afin de prouver les deux implications du théorème 6.6, on construira à partir d'une stratégie gagnante pour Eve dans l'un des deux jeux une stratégie dans le second jeu. Cette stratégie utilise une mémoire qui code une partie dans le premier jeu, où Eve respecte sa stratégie gagnante, et qui est donc une partie gagnante. L'argument principal pour prouver que la nouvelle stratégie est gagnante consiste essentiellement à montrer une correspondance entre les M/B factorisations des deux parties, et conclure en utilisant le fait que la première partie est gagnante.

Implication directe

Supposons que la configuration $(p_{in},\bot)$ soit gagnante dans le jeu $\mathbb{G}$ et considérons une stratégie Φ gagnante pour Eve dans $\mathbb{G}$ depuis $(p_{in},\bot)$.

On définit, à partir de Φ, une stratégie φ pour Eve dans $\widetilde{\mathbb{G}}$ depuis $(p_{in},\bot,\emptyset)$. La stratégie utilise une mémoire qui contient une partie partielle dans $\mathbb{G}$, c'est-à-dire un élément de V^*, et son contenu sera noté Λ. Au départ, Λ est réduit à la configuration $(p_{in},\bot)$. On commence par décrire φ, puis on explique comment Λ est mis à jour. La stratégie φ ainsi que la mise à jour de Λ, sont décrites pour un round.

Choix du coup : On suppose donc que l'on est dans une configuration de la forme (p,γ,R) avec $p \in Q_{\mathbf{E}}$. Le coup donné par φ dépend alors de $\Phi(\Lambda)$:

- si $\Phi(\Lambda) = pop(r)$, alors Eve va dans $\#$ (la proposition 6.2 montrera que ce coup est toujours possible).
- si $\Phi(\Lambda) = skip(p')$, alors Eve va dans (p',γ,R).
- si $\Phi(\Lambda) = push(p',\gamma')$, alors Eve va dans (p,γ,R,p',γ').

Dans ce dernier cas (ou lorsque $p \in Q_{\mathbf{A}}$ et qu'Adam est allé en (p,γ,R,p',γ')), nous devons également expliquer comment Eve joue depuis (p,γ,R,p',γ'). Elle doit choisir un sous-ensemble $R' \subseteq Q$ qui décrit les états atteignables si γ' est dépilé. Afin de définir R', Eve considère l'ensemble des parties possibles dans $\mathbb{G}$, qui commencent par $\Lambda \cdot (p',\gamma'\gamma\sigma)$, si $(p,\gamma\sigma)$ désigne le dernier sommet de Λ, et dans lesquelles elle respecte sa stratégie Φ. Pour chacune de ces parties, Eve regarde si une configuration de la forme $(p'',\gamma\sigma)$ apparaît après $\Lambda \cdot (p',\gamma'\gamma\sigma)$, c'est-à-dire si γ' est un jour dépilé. Si tel est le cas, Eve considère la première configuration $(p'',\gamma\sigma)$ de cette forme qui apparaît après $\Lambda \cdot (p',\gamma'\gamma\sigma)$. Le sous-ensemble R' est exactement l'ensemble des $p'' \in Q$ tel que l'on ait la propriété précédente.

Mise à jour de Λ : La mise à jour de Λ est effectuée à chaque fois que l'on arrive dans un sommet de la forme (p,γ,R). On a alors trois cas, selon la nature du dernier round :

- le round est une bosse triviale, et l'on vient donc de simuler une action $skip(p')$. Si l'on appelle $(p,\gamma\sigma)$ le dernier sommet de Λ, on augmente Λ du sommet $(p',\gamma\sigma)$.
- le round est une bosse, et l'on vient donc de simuler une bosse commençant par une action $push(p',\gamma')$ et se terminant dans un état p''. Si l'on appelle $(p,\gamma\sigma)$ le dernier sommet de Λ, on met à jour Λ en y ajoutant $(p',\gamma'\gamma\sigma)$ suivi d'une série de coups, où Eve respecte Φ, et qui conduisent à dépiler γ' puis à arriver dans $(p'',\gamma\sigma)$.
- le round est une marche et l'on vient donc de simuler une action $push(p',\gamma')$. Si l'on appelle $(p,\gamma\sigma)$ le dernier sommet de Λ, on augmente Λ du sommet $(p',\gamma'\gamma\sigma)$.

Dès lors, à toute partie partielle λ dans $\widetilde{\mathbb{G}}$, où Eve respecte sa stratégie φ, est associée une partie partielle Λ dans $\mathbb{G}$. Une récurrence immédiate permet de prouver que Λ est une partie dans laquelle Eve respecte Φ. Le même raisonnement peut être mené lorsque λ est une partie infinie et la partie Λ associée est alors infinie et commence en $(p_{in},\bot)$. Lors de celle-ci, Eve respecte sa stratégie gagnante Φ. En particulier, la pile explose strictement et sa limite est dans $L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_{n-1} \rhd \mathcal{A}_n)$.

La proposition suivante découle directement de la définition de φ

Proposition 6.2 *Soit λ une partie partielle dans $\widetilde{\mathbb{G}}$ commençant en $(p_{in},\bot,\emptyset)$, se terminant dans un sommet (p,γ,R), et où Eve respecte φ. Soit Λ la partie associée à λ construite par φ. On a alors les points suivants :*

- *Λ se termine dans un sommet de la forme $(p,\gamma\sigma)$ pour un certain $\sigma \in \Gamma^*$.*
- *si la partie Λ est prolongée en une partie où Eve respecte Φ, alors si γ est un jour dépilé, la configuration (r,σ) alors atteinte est telle que $r \in R$.*

La proposition 6.2 implique en particulier que λ n'atteint jamais le sommet *ff*. En effet, un tel coup ne peut être joué que par Adam et fait suite à la simulation d'une règle de dépilement, ce qui contredit la proposition précédente. Dès lors, les parties finies dans λ sont remportées par Eve.

Le cas des parties finies étant réglé, considérons une partie infinie $\lambda = \lambda_1\lambda_2\cdots$ dans $\widetilde{\mathbb{G}}$, commençant en $(p_{in},\bot,\emptyset)$ et dans laquelle Eve respecte φ. Considérons la partie infinie $\Lambda = \Lambda_1\Lambda_2\cdots$, construite par φ lors de la partie λ. La partie Λ est remportée par Eve puisque celle-ci y respecte sa stratégie gagnante φ. Par ailleurs, on a sans difficulté le résultat suivant.

Lemme 6.5 *Notons $Steps_\Lambda = \{n_0 < n_1 < n_2 < \cdots\}$ et $Steps_\lambda = \{m_0 < m_1 < m_2 < \cdots\}$. Pour tout indice $i \geq 0$, on a :*

- *$\Lambda_{n_i} = (p,\gamma\sigma)$ pour un état $p \in Q$, $\gamma \in \Gamma$ et $\sigma \in \Gamma^*$ si et seulement si $\lambda_{m_i} = (p,\gamma,R)$ pour un sous-ensemble $R \subseteq Q$.*
- *$|\Lambda_{n_i}| = |\Lambda_{n_{i+1}}|$ si et seulement si dans le facteur $\lambda_{m_i}\cdots\lambda_{m_{i+1}}$ de λ, tous les arcs sont étiquetés par le mot vide ε.*
- *$|\Lambda_{n_i}| + 1 = |\Lambda_{n_{i+1}}|$ si et seulement si dans le facteur $\lambda_{m_i}\cdots\lambda_{m_{i+1}}$ de λ, tous les arcs sont étiquetés par ε, sauf un qui est étiqueté par γ, où γ est tel que $\Lambda_{n_i} = (p,\gamma\sigma)$ pour un état $p \in Q$, et un contenu de pile $\sigma \in \Gamma^*$.*

Maintenant, comme Λ est une partie infinie commençant en $(p_{in},\bot)$, et dans laquelle Eve respecte sa stratégie gagnante Φ, la pile de $\mathcal{P}$ explose strictement et de plus $\mathrm{StLim}(\Lambda) \in L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1})$. Le lemme 6.5 nous permet alors de conclure que $\mathrm{Lab}(\lambda) = \mathrm{StLim}(\Lambda)$ et donc, $\mathrm{Lab}(\lambda) \in L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}) = \Omega^{ext}_{\mathcal{A}_1\rhd\cdots\rhd\mathcal{A}_n\rhd\mathcal{A}_{n+1}}$, ce qui signifie que λ est remportée par Eve.

Implication réciproque

On suppose maintenant qu'Eve possède une stratégie gagnante φ dans $\widetilde{\mathbb{G}}$ depuis $(p_{in},\bot,\emptyset)$. A partir de φ, on construit une stratégie Φ dans $\mathbb{G}$ depuis $(p_{in},\bot)$.

A la différence de la preuve donnée pour le théorème 5.1 pour les conditions de parité, la stratégie φ ne peut plus être choisie positionnelle et du coup, Φ est un peu plus compliquée. La stratégie Φ est à nouveau une stratégie à pile, mais, à la différence du cas des conditions de parité, celle-ci stocke dans sa pile de stratégie Π non seulement les sommets principaux d'une partie dans $\widetilde{\mathbb{G}}$ (c'est-à-dire une version compacte de la partie), mais la description complète d'une partie (dans la terminologie utilisée dans la preuve du théorème 5.1, on stocke directement l'image par μ de la pile compactée).

On rappelle qu'une partie dans $\widetilde{\mathbb{G}}$, c'est-à-dire un chemin dans $\widetilde{\mathcal{G}}$, est vu comme une suite de sommets dans laquelle on intercale des étiquette prise dans l'alphabet Γ. Par exemple, au lieu d'écrire $v(v,\gamma,v')(v',\varepsilon,v'')$, on écrit $v\gamma v'v''$ pour représenter le chemin commençant en v, allant de v à v' en empruntant un arc étiqueté par γ et enfin, allant de v' à v'' par un arc non étiqueté (c'est-à-dire d'étiquette ε).

Ainsi, l'alphabet de pile Π est l'union des sommets de $\widetilde{\mathcal{G}}$ avec Γ. Dans la suite, $top(\Pi)$ sera le sommet de la pile de stratégie Π. Par $StCont(\Pi)$, nous désignons le mot obtenu en lisant la pile de stratégie Π de bas en haut (en omettant le symbole de fond de pile). Dans une partie où Eve respecte Φ, $StCont(\Pi)$ sera une partie dans $\widetilde{\mathbb{G}}$ commençant en $(p_{in},\bot,\emptyset)$, où Eve respecte sa stratégie gagnante φ. De plus, lors d'une partie Λ où Eve respecte Φ, nous aurons à tout moment que $top(\Pi) = (p,\gamma,R)$ et $\mathrm{Lab}(StCont(\Pi))) = \sigma$ si et seulement si la position courante dans Λ est $(p,\gamma\sigma)$. Enfin, si Eve respecte Φ, et si γ est un jour dépilé, la configuration alors atteinte sera de la forme (r,σ) avec $r \in R$.

Afin de décrire Φ, on suppose que l'on se trouve dans une configuration $(p,\gamma\sigma)$, et que $top(\Pi) = (p,\gamma,R)$. Dans un premier temps, nous décrivons comment Eve joue si $p \in Q_{\mathbf{E}}$, et nous expliquons ensuite comment la pile de stratégie est mise à jour.

- **Choix du coup :** Supposons que $p \in Q_{\mathbf{E}}$ et donc qu'Eve doive jouer depuis $(p,\gamma\sigma)$. Pour cela, Eve considère la valeur de φ sur $StCont(\Pi)$.
  S'il s'agit d'un coup vers $\sharp t$, Eve applique une transition $pop(r)$ pour un état $r \in R$ (le lemme 6.6 montrera qu'un tel état r existe).
  Si le coup donné par φ est d'aller vers un sommet (p',γ,R), Eve applique la transition $skip(p')$.
  Si le coup donné par φ est d'aller vers un sommet (p,γ,R,p',γ'), alors Eve applique la transition $push(p',\gamma')$.
- **Mise à jour de Π :** Si le coup, joué par Eve ou Adam, a été d'aller de $(p,\gamma\sigma)$ vers une configuration (r,σ), Eve met à jour Π de la façon suivante : elle dépile dans Π jusqu'à trouver une lettre $\gamma' \in \Gamma$, qu'elle dépile également. Enfin, elle empile (r,γ',R'). Ce cas est illustré par la figure 6.4.
  Si le coup, joué par Eve ou Adam, a consisté à aller de $(p,\gamma\sigma)$ vers une configuration $(p',\gamma\sigma)$, Eve met à jour Π en empilant (p',γ,R).
  Si le coup, joué par Eve ou Adam, a consisté à aller de $(p,\gamma\sigma)$ vers une configuration $(p',\gamma'\gamma\sigma)$, on pose $(p,\gamma,R,p',\gamma',R') = \varphi(StCont(\Pi)\cdot(p,\gamma,R,p',\gamma'))$. Intuitivement, R' est l'ensemble des états qu'Eve peut assurer si γ' est un jour dépilé. Eve met à jour Π en empilant successivement (p,γ,R,p',γ'), $(p,\gamma,R,p',\gamma',R')$, γ (on décrit une marche donc on passe par un arc étiqueté) et (p',γ',R').

La stratégie Φ ainsi définie utilise une mémoire infinie, à savoir une pile. Cependant, Φ est finiment descriptible même si, à la différence de ce qui se passait dans le cas des conditions étudiées dans le chapitre 5, la hauteur de la pile de stratégie Π peut être arbitrairement grande par rapport à celle de la configuration courante dans le jeu (c'est en fait le cas lorsqu'il y a une infinité de bosse).

Concernant la signification du contenu de Π, on a le résultat suivant.

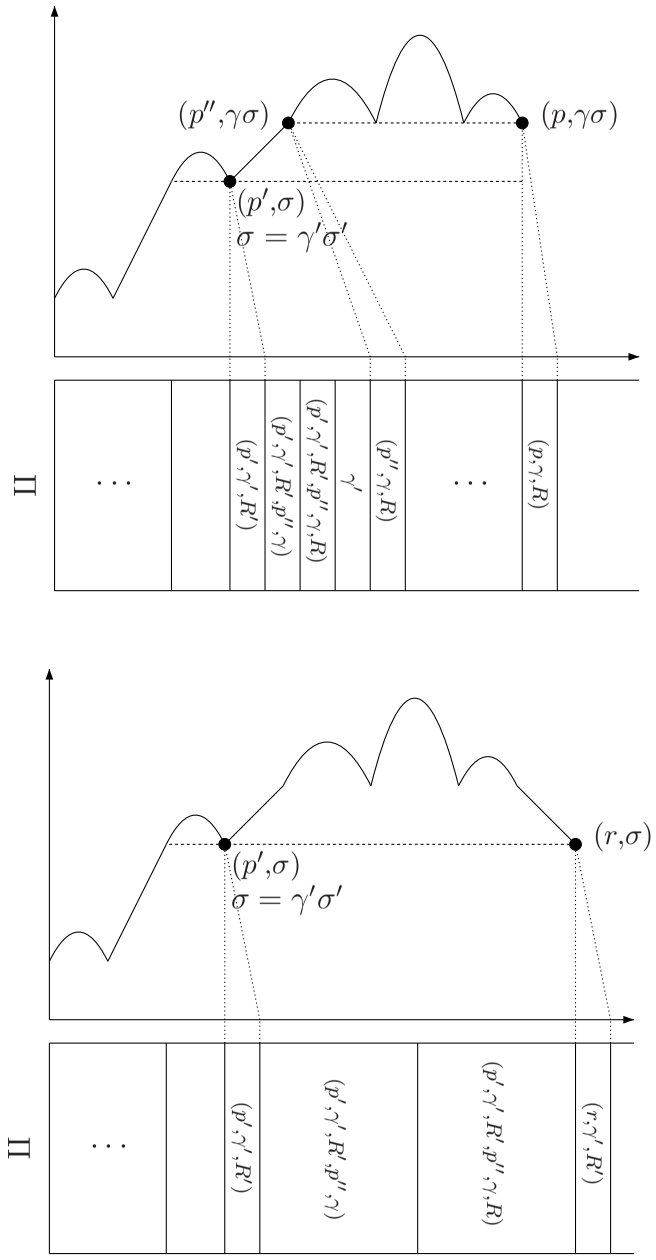


FIG. 6.4 – *Mise à jour de la pile de stratégie* Π

Lemme 6.6 *Soit Λ une partie partielle dans $\mathbb{G}$, où Eve respecte Φ, commençant en $(p_{in},\bot)$ et se terminant dans une configuration $(p,\gamma\sigma)$. On a alors les faits suivants :*

1. *$top(\Pi) = (p,\gamma,R)$ pour un sous-ensemble $R \subseteq Q$.*
2. *$StCont(\Pi)$ est une partie partielle dans $\widetilde{\mathbb{G}}$ commençant en $(p_{in},\bot,\emptyset)$, se terminant en (p,γ,R) et dans laquelle Eve respecte φ.*
3. *$\mathrm{Lab}(StCont(\Pi)) = \sigma$.*
4. *si Λ est étendue par une transition dépilant le sommet de pile γ, la configuration atteinte, (r,σ), est telle que $r \in R$.*

Preuve. La preuve est faite par récurrence sur Λ. On commence par montrer que le dernier point est une conséquence du deuxième. Supposons donc que la transition après $(p,\gamma\sigma)$ soit $pop(r) \in \Delta(p,\gamma)$. Il y a donc un arc dans $\widetilde{\mathcal{G}}$ de (p,γ,R) vers $\#$ ou ff, selon que l'on a $r \in R$ ou $r \notin R$. Si $p \in Q_{\mathbf{E}}$, par définition de Φ, la transition est vers $\#$ et l'on a donc $r \in R$. Si $p \in Q_{\mathbf{A}}$, alors si l'on avait $r \notin R$, la partie $StCont(\Pi)$ pourrait être prolongée en une partie se terminant en ff. Or, ceci n'est pas possible, car d'après le second point, il s'agit d'une partie où Eve respecte sa stratégie gagnante φ.

Supposons maintenant que le résultat est établi pour une partie Λ, et considérons une extension Λ' de Λ. On a deux cas, selon la nature de la transition étendant Λ en Λ' :

- Λ' est obtenu en jouant une règle de type *skip* ou de type *push*. Dans ce cas, le résultat est trivial.
- Λ' est obtenu en jouant une règle de type *pop*. Appelons $(p,\gamma\sigma)$ la dernière configuration de Λ, et soit R la dernière composante de $top(\Pi)$ dans $(p,\gamma\sigma)$. Par hypothèse de récurrence, on a $\Lambda' = \Lambda \cdot (r,\sigma)$ avec $r \in R$. De la manière dont Π est mise à jour, on déduit facilement que la nouvelle pile de stratégie Π a la propriété voulue (on se référera pour cela à la figure 6.4).

□

On peut alors conclure. Considérons une partie infinie Λ dans $\mathbb{G}$, commençant en $(p_{in},\bot)$, et où Eve respecte Φ. La pile dans Λ explose strictement car sinon, on construit aisément une partie, limite des $StCont(\Pi)$ (comme dans les preuves du chapitre 5), dans $\widetilde{\mathbb{G}}$, commençant en $(p_{in},\bot,\emptyset)$, où Eve respecte sa stratégie gagnante φ, et où l'on ne visite qu'un nombre fini d'arcs étiquetés par un mot non vide. On a alors une contradiction avec le fait que φ est une stratégie gagnante pour Eve. Dès lors, la pile explosant strictement, il est facile de voir que sa limite est égale à l'étiquetage d'une partie dans $\widetilde{\mathbb{G}}$, commençant en $(p_{in},\bot,\emptyset)$, et où Eve respecte sa stratégie gagnante φ. Une telle limite appartient donc bien à $L(\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_{n-1} \rhd \mathcal{A}_n)$, ce qui achève la preuve.

Le cas particulier de la condition d'explosion stricte

Nous nous intéressons à nouveau à la condition d'explosion stricte, et nous montrons comment les résultats établis dans le paragraphe 5.3 peuvent être vus comme un corollaire du théorème 6.6.

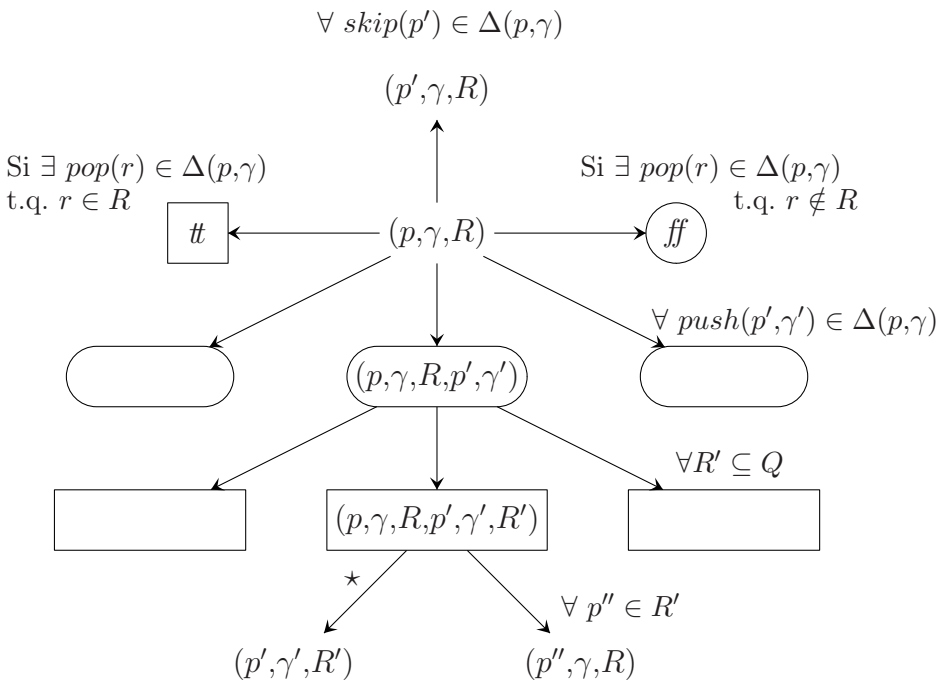


FIG. 6.5 – *Structure locale de $\widetilde{\mathcal{G}}$ pour la condition d'explosion stricte.*

La condition d'explosion stricte est la condition interne $\Omega^{int}_{\mathcal{A}}$ pour tout automate déterministe $\mathcal{A}$ reconnaissant le langage ω-régulier Γ^ω (où Γ désigne l'alphabet de pile sous-jacent au graphe de jeu). Dès lors, on déduit facilement du théorème la décidabilité des jeux équipés d'une condition d'explosion stricte.

Précisons la construction afin d'établir la complexité du problème de décision associé. On considère donc un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$ muni d'une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de son ensemble d'états de contrôle Q. On désigne par $\mathcal{G}$ le graphe de jeu associé, et par $\mathbb{G}$ le jeu d'explosion stricte sur $\mathcal{G}$. Le jeu *équivalent*, joué sur un graphe fini comme décrit dans la réciproque du théorème 6.6, est équipé de la condition de gain $\Omega^{ext}_{\mathcal{A}}$. Dès lors, ce jeu peut être vu comme un jeu de Büchi, car Eve gagne une partie si et seulement si elle visite une infinité d'arcs étiquetés par un mot non vide. Le graphe $\widetilde{\mathcal{G}}$ est donné dans la figure 6.5 (les arcs finaux sont marqués par le symbole $\star$), et si l'on souhaite le transformer en un graphe de jeu équipé d'une condition de Büchi sur les sommets et non pas sur les arcs, on retrouve le graphe de jeu de la figure 5.8.

Du théorème 6.6, on déduit le corollaire suivant.

Corollaire 6.2 *Eve possède une stratégie gagnante dans le jeu d'explosion stricte $\mathbb{G}$ depuis $(p_{in},\bot)$ si et seulement si elle possède une stratégie gagnante dans le jeu de Büchi $\widetilde{\mathbb{G}}$ depuis $(p_{in},\bot,\emptyset)$.*

Concernant la complexité, comme $\mathbb{G}$ est un jeu de Büchi joué sur un graphe de

taille exponentielle, on obtient le corollaire suivant.

Corollaire 6.3 *On peut décider le gagnant dans un jeu sur un graphe de processus à pile équipé d'une condition d'explosion stricte en* DEXPTIME.

6.4 Complexité

Le théorème 6.3 donnait la décidabilité pour un jeu sur un graphe fini muni d'une condition de la forme $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$, et pour un jeu sur un graphe de processus à pile muni d'une condition de la forme $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_n \rhd \mathcal{A}_{n+1}}$. Pour ce faire, on utilisait deux réductions : la première, des jeux sur un graphe fini vers les jeux sur un graphe de processus à pile, et la seconde, des jeux sur un graphe de processus à pile vers les jeux sur un graphe fini. La première transformation est polynomiale, alors que la seconde est exponentielle. A la fin, on doit résoudre un jeu de parité sur un graphe de processus à pile, ce qui demande un temps exponentiel. On en déduit facilement l'analyse de l'algorithme résultant du théorème 6.3

Théorème 6.7 *Pour tout $k \geq 0$, décider le gagnant dans un jeu sur un graphe fini muni d'une condition de gain externe de la forme $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_k \rhd \mathcal{A}_{k+1}}$, est un problème $(k+1)$-*DEXPTIME.

*Pour tout $k \geq 0$, décider le gagnant dans un jeu sur un graphe de processus à pile muni d'une condition de gain interne de la forme $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_k \rhd \mathcal{A}_{k+1}}$, est un problème $(k+2)$-*DEXPTIME.

Concernant les bornes inférieures, on a le résultat suivant pour les jeux sur des graphes finis.

Théorème 6.8 *Pour tout $k \geq 0$, décider le gagnant dans un jeu sur un graphe fini muni d'une condition de gain externe de la forme $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_k \rhd \mathcal{A}_{k+1}}$ est un problème $(k+1)$-*DEXPTIME *dur.*

Compte tenu que, pour tout $k \geq 0$, décider le gagnant dans un jeu sur un graphe fini muni d'une condition de la forme $\Omega^{ext}_{\mathcal{A}_1 \rhd \ldots \rhd \mathcal{A}_{k+1} \rhd \mathcal{A}_{k+2}}$ se réduit polynomialement à décider le gagnant dans un jeu sur un graphe de processus à pile muni d'une condition de la forme $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_k \rhd \mathcal{A}_{k+1}}$, on a le corollaire suivant du théorème 6.8

Corollaire 6.4 *Pour tout $k \geq 0$, décider le gagnant dans un jeu sur un graphe de processus à pile muni d'une condition de gain interne de la forme $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_k \rhd \mathcal{A}_{k+1}}$, est un problème $(k+2)$-*DEXPTIME *dur.*

Les théorèmes 6.7 et 6.8 et le corollaire 6.4 impliquent le corollaire suivant.

Corollaire 6.5 *Décider le gagnant dans un jeu sur un graphe fini muni d'une condition de gain interne de la forme $\Omega^{ext}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_k \rhd \mathcal{A}_{k+1}}$ est un problème non élémentaire qui est* ELEMENTARY*-dur.*

Décider le gagnant dans un jeu sur un graphe de processus à pile muni d'une condition de gain interne de la forme $\Omega^{int}_{\mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_k \rhd \mathcal{A}_{k+1}}$ est un problème non élémentaire qui est ELEMENTARY*-dur.*

6.4.1 Preuve du théorème 6.8

La preuve commence par le cas particulier $k = 0$, dont le traitement est différent des autres cas. Comme la preuve du théorème 6.8 est très technique, nous traitons à part le cas $k = 1$. En effet, ce cas est un peu plus simple que le cas général qui peut être alors vu comme une généralisation de ce dernier.

La preuve repose sur une simulation d'une machine de Turing utilisant un espace $tow(k,N)$ pour un entier N polynomial dans la taille de l'entrée. Une telle machine de Turing est équivalente à une machine de Turing déterministe utilisant un temps $tow(k+1,N)$ [25, 48, 24].

Dans toutes les preuves, l'automate à pile déterministe $\mathcal{A}_{k+1}$ ne sera pas équipé d'une condition de parité, mais d'une condition plus faible, à savoir une condition d'accessibilité.

Le cas particulier où $k = 0$

Dans [83, 84], I. Walukiewicz montre que décider le gagnant dans un jeu d'accessibilité sur un graphe de processus à pile est DEXPTIME-dur. Pour cela, il simule, par un jeu d'accessibilité sur un graphe de processus à pile, une machine de Turing alternante d'espace linéaire. Le jeu se déroule de la façon suivante : Eve empile des symboles décrivant des configurations de la machine de Turing que l'on souhaite simuler. Après chaque description d'une configuration, selon que l'état de contrôle dans celle-ci est existentiel ou universel, Eve ou Adam choisit une transition de la machine de Turing. Ensuite, Eve empile la description de la nouvelle configuration de la machine de Turing. La simulation se termine si Eve décrit un jour une configuration finale, et dans ce cas, elle remporte la partie. Afin de forcer Eve à décrire un calcul valide de la machine de Turing, Adam peut, après chaque description d'une configuration, vérifier la validité de celle-ci relativement à la configuration précédente et à la dernière transition de la machine de Turing simulée. Pour cela, il choisit une case à vérifier (il y en a un nombre linéaire), et dépile pour vérifier la validité de cette dernière. Si la case était correcte, Eve remporte la partie, sinon c'est Adam qui gagne.

Ainsi, dans cette simulation, la pile n'est véritablement utilisée que pour stocker de l'information (ce qui est fait en empilant des symboles). On ne dépile que pour vérifier la validité d'une configuration et alors la partie se termine. Il n'est pas difficile de voir que l'on peut faire une réduction similaire en utilisant un jeu, sur un graphe fini, équipé d'une condition de la forme $\Omega_{\mathcal{A}_1}$. Pour cela, on considère un graphe de jeu (schématisé dans la figure 6.6) formé d'une composante faite de sommets appartenant à Eve, et desquels partent des transitions étiquetées par les symboles permettant de décrire des configurations de la machine de Turing. Eve peut sortir de cette composante (lorsqu'elle a fini de décrire une configuration), et selon la nature de la dernière (existentielle ou universelle), on arrive dans un sommet, sommet, appartenant à Eve ou à Adam, dans lequel est codé l'état de contrôle et la case sous la tête de lecture de la dernière configuration de la machine de Turing. De là, le joueur sommet *Check*, appartenant à Adam, via un arc étiqueté par une transition de la machine de Turing (on note Δ_M la relation de transition de la machine). Depuis aller dans un sommet *Contest* dans lequel la partie boucle

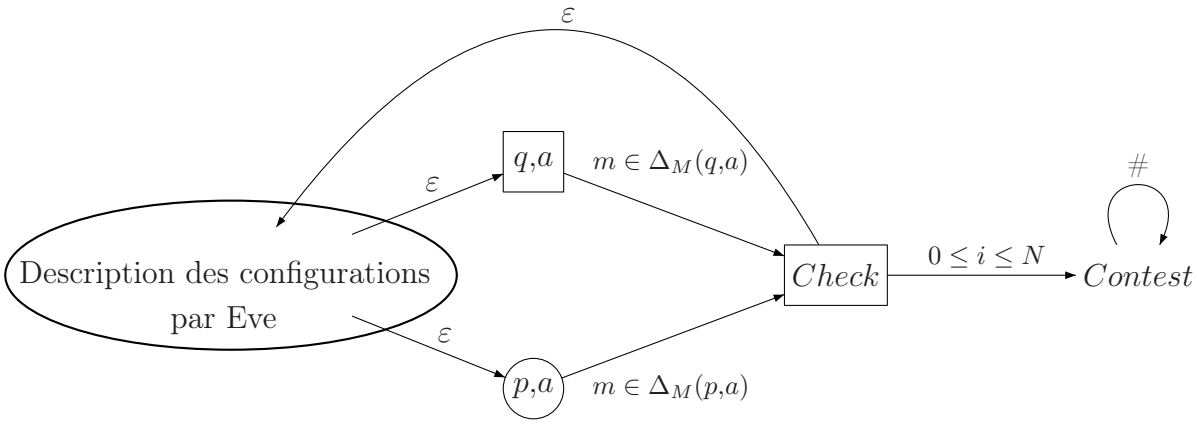


FIG. 6.6 – *Graphe de jeu pour* $k = 0$

la partie boucle infiniment, via un arc étiqueté par un entier i (borné par la taille N des configurations de la machine de Turing), indiquant ainsi qu'il pense qu'il y a une erreur dans la description de la i-ème case de la dernière configuration, sinon, il revient dans la composante où Eve décrit les configurations.

L'automate $\mathcal{A}_1$ vérifie que la première configuration décrite par Eve est bien la configuration initiale. Il accepte s'il voit une configuration finale. Enfin, s'il lit un entier i, il dépile afin de vérifier la validité du i-ème symbole de la dernière configuration décrite par Eve. En cas d'erreur de la part de cette dernière, il boucle dans un état non final, sinon il accepte.

Le caractère polynomial de la réduction provient du fait que la machine de Turing simulée est d'espace linéaire. La validité de la réduction ne pose pas de problème particulier.

Le cas particulier où $k = 1$

On se concentre maintenant sur le cas où $k = 1$. On souhaite donc simuler une machine de Turing alternante M, utilisant un espace de taille 2^N, où N est polynomial dans la taille de M. La simulation est faite par un jeu sur un graphe fini équipé d'une condition de la forme $\Omega^{ext}_{\mathcal{A}_1 \triangleright \mathcal{A}_2}$. Plus précisément, Eve possède une stratégie gagnante depuis un sommet initial dans le jeu si et seulement si M accepte depuis son état initial i la configuration réduite à des symboles blancs.

On pourrait prendre la même simulation que précédemment. Cependant, lorsque depuis le sommet *Check*, Adam voudrait contester le contenu d'une case, deux problèmes se posent. Adam doit indiquer l'indice de la case, qui peut alors être exponentiel. L'automate vérifiant la contestation peut avoir à dépiler un nombre exponentiel de symboles et ne peut, s'il est de taille polynomiale, compter un tel nombre de symboles.

La solution passe par un codage en binaire des indices des cases de la machine. On adopte donc la représentation suivante d'une configuration $a_0 a_1 \cdots a_{2^N-1}$ de M :

$$(\#a_0a_1)(n_0,\widetilde{n}_0)^{2^N}(a_0a_1a_2)(n_1,\widetilde{n}_1)^{2^N}\ldots(a_{2^N-2}a_{2^N-1}\#)(n_{2^N-1},\widetilde{n}_{2^N-1})^{2^N}$$

où n_i est la représentation en binaire de i sur N bits avec le bit de poids fort à gauche et $\widetilde{n}_i$ est donc la représentation de i sur N bits avec le bit de poids fort à droite. Par exemple, si $N = 4$, $n_{10} = 1010$ et $\widetilde{n}_{10} = 0101$. Le couple $(n_i,\widetilde{n}_i)$ représente la suite de paire de bits $(n_i^0,n_i^{N-1})(n_i^1,n_i^{N-2})\dots(n_i^{N-1},n_i^0)$ où n_i^j est le bit d'indice j de n_i lu de gauche à droite. Par exemple, $(n_{10},\widetilde{n}_{10}) = (1,0)(0,1)(1,0)(0,1)$. On rappelle que, pour un mot u, u^{2^N} est le mot obtenu en itérant 2^N fois u. Par exemple, $u^4 = u \cdot u \cdot u \cdot u$. La suite $(n_i,\widetilde{n}_i)$ est appelée un *compteur*. La valeur d'un compteur est itérée 2^N fois afin de permettre une recherche brutale d'une position lors d'une contestation d'Adam, où l'on va dépiler à la recherche d'un compteur. Pour cela, on testera tous les compteurs stockés dans la pile avec le compteur itéré. Le fait que le compteur soit itéré 2^N fois permet de tester son égalité avec toutes les valeurs possibles (qui sont au nombre de 2^N). En fait, comme $k = 1$, on pourrait ne pas itérer les compteurs, mais la généralisation pour $k > 1$ ne serait plus aussi naturelle.

On ne décrit pas de façon formelle le graphe de jeu, mais on se contente de décrire le déroulement d'une partie. L'existence d'un graphe de jeu de taille polynomiale permettant d'obtenir de telle partie ne pose pas de problème, même si une description précise serait extrêmement technique. Une partie se déroule de la façon suivante : Eve décrit la configuration initiale, c'est-à-dire l'état initial suivi de $2^N - 1$ symboles blancs. Ensuite, selon que l'état de la configuration décrite est existentiel ou universel, Eve ou Adam, choisit une transition valide de M depuis cette configuration. Eve décrit ensuite la configuration successeur et ainsi de suite. Si on atteint un jour une configuration finale, la partie va dans un sommet spécial, où elle boucle via un arc étiqueté par un symbole spécial, ♡.

Afin de forcer Eve à donner une description correcte des configurations, il y a cinq symboles $!_c,\widetilde{!_c},!_v,\widetilde{!_v}$ (c signifie *copie* et v *valeur*), et ?. Adam peut les écrire pour contester la validité de la dernière lettre écrite par Eve. Ils peuvent être utilisés dans les cas suivants :

- la j-ème paire de bit d'un compteur $(n_i,\widetilde{n}_i)$ est contestée. S'il s'agit de la première copie du compteur, l'erreur concerne la non-égalité de la valeur du compteur par rapport à celle du compteur précédent augmenté de 1. Dans ce cas, Adam écrit $!_v$ (si l'erreur concerne n_i) ou $\widetilde{!_v}$ (si l'erreur concerne $\widetilde{n}_i$) juste après la paire incriminée.
  S'il s'agit d'une des itérations, Adam écrit $!_c$ (si l'erreur concerne n_i) ou $\widetilde{!_c}$ (si l'erreur concerne $\widetilde{n}_i$) juste après la paire incriminée.
- la valeur d'un triplet $(a_{i-1}a_ia_{i+1})$ est contestée. Dans ce cas, Adam écrit ? juste après le triplet incriminé.

Une fois qu'une contestation $!_c$, $\widetilde{!_c}$, $!_v$ ou $\widetilde{!_v}$ est effectuée, la partie boucle sur un sommets spécial via un arc étiqueté par un symbole spécial, ♠. Dans le cas d'une contestation ?, l'itération du prochain compteur est écrite (et peut être contestée), et la partie boucle ensuite sur le sommet précédent via l'arc étiqueté par ♠.

L'alternance des coups entre les joueurs (permettant à Adam de contester), la mémoire pour stocker l'état et la lettre sous la tête de lecture de la dernière configuration, l'existence d'une configuration finale, sont codés dans le graphe de jeu. Il en va de même pour obliger Eve à changer de composante une fois la configuration

écrite, pour la forcer à commencer par décrire la configuration initiale ou pour l'obliger à écrire $(n_0,\widetilde{n_0})$ comme premier compteur. De même, Eve doit écrire un triplet de la forme (a_i,a_{i+1},a_{i+2}) si le dernier triplet de lettres était (a_{i-1},a_i,a_{i+1}). On peut se convaincre assez facilement que toutes ces contraintes peuvent se coder dans un graphe de jeu de taille polynomiale en la taille de la machine de Turing M.

Evoquons maintenant comment la validité d'une contestation est décidée. Pour une contestation concernant un compteur, il faut regarder la valeur du compteur précédent et la confronter à la valeur contestée. Pour ce qui est d'une contestation ? concernant un triplet de lettres, on doit retrouver le triplet correspondant dans la configuration précédente, ainsi que la dernière transition de la machine. Ensuite la vérification est simple. Afin de retrouver le triplet de même indice dans la configuration précédente, on utilise les compteurs (les deux triplets ont le même compteur). C'est pour cette raison que, suite à une contestation de la forme ?, on écrit le compteur associé avant de boucler. C'est aussi pour cette raison que l'on doit être certain de la validité des compteurs avant de vérifier la validité des triplets. Dès lors, dans la condition de gain $\Omega^{ext}_{\mathcal{A}_1 \triangleright \mathcal{A}_2}$, $\mathcal{A}_1$ va servir à vérifier la validité des compteurs et ensuite $\mathcal{A}_2$ vérifie la validité des triplets.

L'automate $\mathcal{A}_1$ copie le mot qu'il lit dans sa pile. S'il trouve une contestation $!_v$ ou $\widetilde{!_v}$, il détermine l'indice j du bit incriminé (qui est borné par N). Pour cela, il dépile et compte le nombre de symboles dépilés avant de trouver un triplet de lettres. Ce nombre est égal à j. Ensuite, si la contestation est $!_v$, il dépile les j paires suivantes et considère leur seconde composante pour calculer la valeur du j-ème bit du compteur obtenu en rajoutant 1 au compteur décrit dans la pile. Il détermine alors si la contestation était valide ou non. La figure 6.4.1 illustre ce cas ainsi que le suivant. Si la contestation est $\widetilde{!_v}$, il dépile les $(N-j-1)$ paires suivantes, et ensuite considère les premières composantes des j paires suivantes pour déterminer si la contestation était valide ou non. Une fois la validité de la contestation établie, $\mathcal{A}_1$ ignore le reste du mot, et empile infiniment le symbole ♠ si la contestation était valide (Adam avait raison), et le symbole ♡ sinon (Eve avait raison).

Si la contestation est $!_c$ ou $\widetilde{!_c}$, il suffit de dépiler les N dernières paires de bit de la pile, de comparer le nouveau sommet avec la paire contestée, et de tester alors l'égalité. La figure 6.4.1 illustre ce cas. Comme précédemment, une fois la contestation testée, $\mathcal{A}_1$ empile infiniment le symbole ♠ si elle était valide, et le symbole ♡ sinon.

L'automate $\mathcal{A}_2$ copie le mot qu'il lit en entrée dans sa pile. Il va dans un état final s'il lit le symbole ♡ (et donc accepte), et dans un état bloquant et non final, s'il lit le symbole ♠ (et donc rejette). Remarquons que la pile de $\mathcal{A}_1$ explose toujours strictement et ne peut contenir qu'une contestation ?. On explique maintenant le comportement de $\mathcal{A}_1$ quand lit ?. Le raisonnement est illustré dans la figure 6.4.1. L'automate $\mathcal{A}_1$ dépile jusqu'à trouver la description d'une transition de la machine de Turing. Ensuite, il lit la première copie du compteur du mot d'entrée en même temps que le compteur qui se trouve dans sa pile. Il les compare (il compare la première composante du premier avec la seconde du second). S'ils sont égaux, on est sur la bonne case, $\mathcal{A}_2$ regarde le triplet suivant dans sa pile et regarde si la transition était correcte et conclut donc quant à la validité de la contestation. Sinon, il dépile jusqu'à trouver un triplet et recommence le même test avec le nouveau compteur,

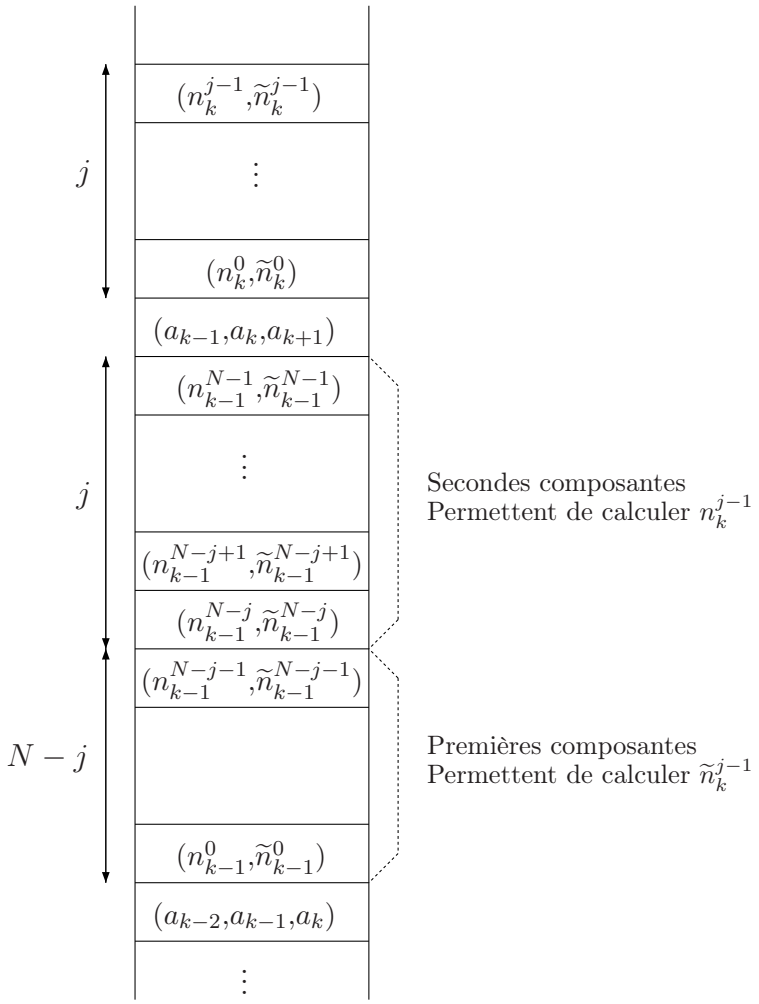


FIG. 6.7 – *Contenu de la pile de $\mathcal{A}_1$ au moment de la lecture de $!_v$*

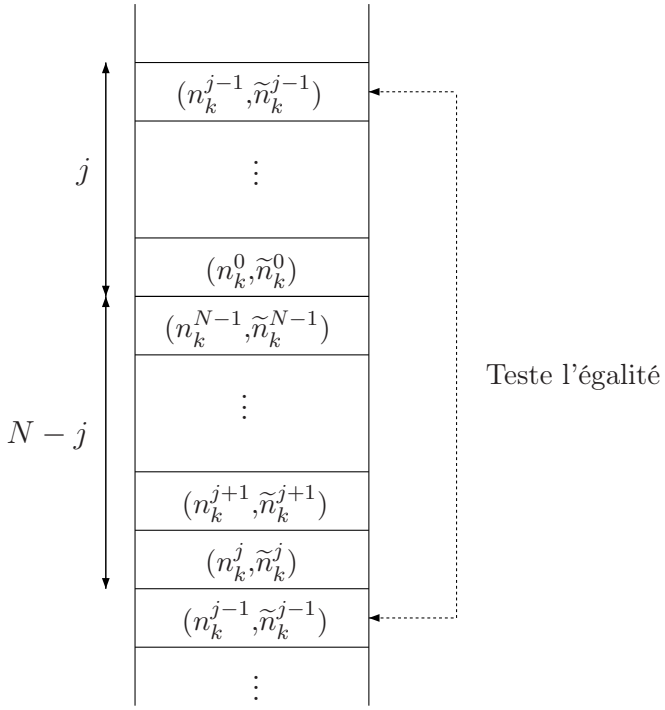


FIG. 6.8 – *Contenu de la pile de $\mathcal{A}_1$ au moment de la lecture de $!_c$*

et ainsi de suite. La procédure se termine puisque tous les compteurs sont présents dans la pile et qu'il y a 2^N itérations du compteur recherché. L'automate $\mathcal{A}_2$ peut donc décider de la validité de la contestation. Si la contestation était valide (Adam avait raison), le mot est rejeté, sinon (Eve avait raison), le mot est accepté.

On se convainc que la réduction est polynomiale et qu'Eve possède une stratégie gagnante si et seulement si l'entrée vide est accepté par M. Une stratégie gagnante pour Eve consiste alors à décrire un calcul acceptant. Si M rejette, une stratégie gagnante pour Adam, consiste à décrire un calcul rejetant et à contester si Eve se trompe dans la description.

Remarque 6.1 Dans les comportements de $\mathcal{A}_1$ et $\mathcal{A}_2$, on peut objecter que les automates dépilent parfois alors qu'ils ne lisent pas l'entrée. C'est en particulier le cas quand ils recherchent un compteur dans la pile. Cependant, on peut modifier la réduction de telle sorte qu'Adam écrive une lettre assimilée à un blanc, qui sera lue lorsque les automates dépilent sans lire de lettre. C'est alors à Adam d'écrire suffisamment de blancs.

Le cas général : $k \geq 1$

Nous nous intéressons maintenant au cas général. Afin de simuler une machine de Turing alternante utilisant un espace $tow(k,N)$ par un jeu sur un graphe fini muni d'une condition de gain de la forme $\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_k \triangleright \mathcal{A}_{k+1}}$, on généralise la construction pour $k = 1$.

Pour passer du cas $k = 0$ au cas $k = 1$, nous avions dû raffiner la représentation des configurations de la machine de Turing en insérant des compteurs binaires de

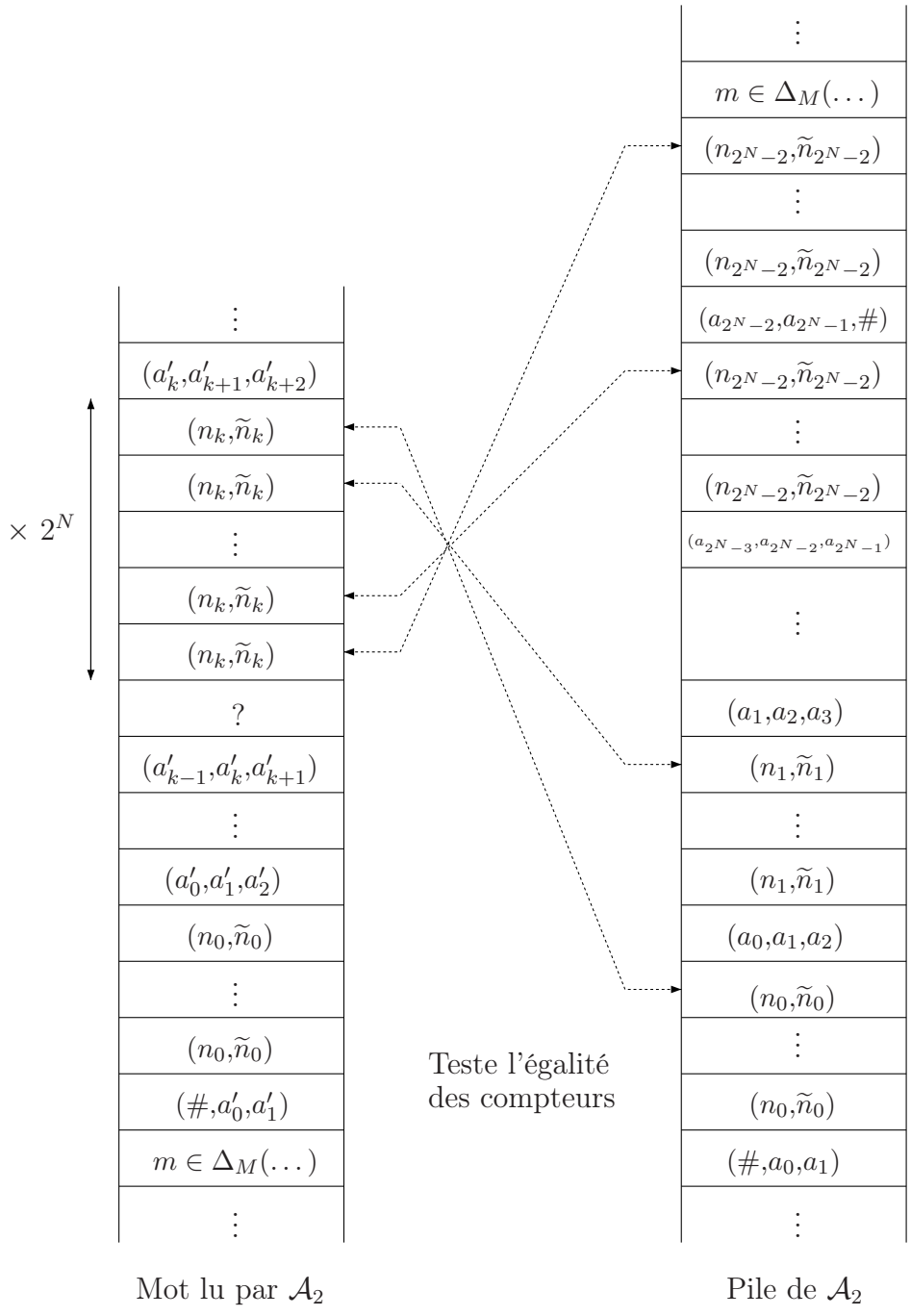


FIG. 6.9 – *Comportement de $\mathcal{A}_2$ au moment de la lecture de ?*

tailles linéaires permettant de représenter les indices (exponentiels) des cases de la machine de Turing. Cette fois, les compteurs ne peuvent plus être linéaires et, si l'on garde le même codage, ces derniers ne pourrons donc plus être vérifiés. L'idée naturelle est donc d'insérer des compteurs au sein des compteurs pour pouvoir les vérifier. Et récursivement, jusqu'à avoir de petits compteurs, c'est-à-dire des compteurs de taille linéaire. Ce sont ces compteurs qui seront vérifiés par le premier automate, puis le second automate vérifiera, à l'aide de ces compteurs, les compteurs de taille exponentielle, puis le troisième vérifiera les compteurs de taille doublement exponentielle, et ainsi de suite. Enfin, une fois les gros compteurs validés, le dernier automate vérifiera les triplets de lettres.

Plus formellement, on définit la notion de h-exp décomposition d'un entier.

Définition 6.4 (h-exp décomposition) *La* 1-exp décomposition *d'un entier* $0 \leq R < tow(1,N)$ *est la suite* $((b_{N-1},b_0)\dots(b_1,b_{N-2})(b_0,b_{N-1}))^{tow(1,N)}$, *où* $b_{N-1}b_{N-2}\dots b_1b_0$ *est la représentation binaire de* R *avec le bit de poids fort à gauche, et* $b_0b_1\dots b_{N-2}b_{N-1}$ *est la représentation de* R *avec le bit de poids fort à droite.*

Soit $h > 1$ *et soit* R *un entier tel que* $R < tow(h,N)$. *La* h*-exp décomposition de* R *est la suite :*

$$\Big[(b_0,b_{tow(h-1,N)-1})\alpha_0(b_1,b_{tow(h-1,N)-2})\alpha_1\cdots(b_{tow(h-1,N)-1},b_0)\alpha_{tow(h-1,N)-1}\Big]^{tow(h,N)}$$

où l'on a :

- $b_{tow(h-1,N)-1}\dots b_1b_0$ *est la représentation binaire de* R *avec le bit de poids fort à gauche.*
- $b_0\dots b_{tow(h-1,N)-2}b_{tow(h-1,N)-1}$ *est la représentation binaire de* R *avec le bit de poids fort à droite.*
- α_i *est la* $(h-1)$*-exp décomposition de* i.

Pour des raisons pratiques, nous supposons que les alphabets binaires utilisés dans les h*-exp décompositions sont différents, pour tout niveau* h. *Les décompositions sont alors facilement lisibles.*

Enfin, on désigne une suite $(b_0,b_{tow(h-1,N)-1})\dots(b_{tow(h-1,N)-1},b_0)$ *comme un* compteur de niveau h.

Une configuration $a_0a_1\cdots a_{tow(k,N)-1}$ de la machine de Turing M sera alors représentée par un mot :

$$(\#a_0a_1)n_0(a_0a_1a_2)n_1\dots(a_{(tow(k,N)-2)}a_{(tow(k,N)-1)}\#)n_{tow(k,N)-1},$$

où n_i est la k-exp décomposition de i. En particulier, pour $k = 1$, on retrouve exactement le représentation utilisée dans le paragraphe 6.4.1.

Une partie se déroule alors comme dans les cas précédents : Eve commence par décrire la configuration initiale, puis, selon que l'état de contrôle dans cette dernière est existentiel ou universel, Eve ou Adam choisit une transition, puis Eve décrit la nouvelle configuration, et ainsi de suite. Si à un moment une configuration finale est atteinte, la partie boucle via un arc étiqueté par $\heartsuit$.

Comme précédemment, pour empêcher Eve de tricher, Adam peut contester n'importe quel symbole. Du fait qu'il y a plusieurs niveau de compteurs, les symboles

pour contester sont maintenant $!_c^h$,$\widetilde{!_c^h}$,$!_v^h$,$\widetilde{!_v^h}$ pour tout $h = 1 \dots k$ (c est pour *copie* et v pour *valeur*), et ?. Ils peuvent être joués dans les cas suivants :

- la j-ème paire de bit d'un compteur de niveau h est fausse. S'il s'agit de la première copie, l'erreur concerne la valeur par rapport au compteur précédent. Dans ce cas, Adam joue, juste après la paire de bit incriminée, la contestation $!_v^h$, si l'erreur concerne la représentation avec le bit de poids fort à gauche, et $\widetilde{!_v^h}$ sinon. Si l'erreur concerne une valeur suivante, c'est-à-dire que l'erreur concerne l'itération, Adam joue $!_c^h$ ou $\widetilde{!_c^h}$, selon la représentation fautive.
- s'il conteste un triplet (a_{i-1},a_i,a_{i+1}), Adam joue ? juste après.

Lorsqu'une contestation sur un compteur de niveau 1 a été jouée, la partie va dans un sommet où elle boucle via un arc étiqueté par un symbole spécial ♠. Dans le cas d'une contestation pour un compteur de niveau $h > 1$, l'itération de la $(h-1)$-exp décomposition de l'indice du bit incriminé est écrite, et peut être contestée, et la partie boucle ensuite via un arc étiqueté par le symbole spécial ♠. Enfin, lorsqu'une contestation ? est jouée, l'itération de la k-exp décomposition de l'indice de la case incriminée est écrite, et la partie boucle ensuite via un arc étiqueté par le symbole spécial ♠.

Comme précédemment, les contraintes d'alternance des coups, la mémoire, l'obligation pour Eve de changer de composante une fois la configuration écrite, de commencer par décrire la configuration initiale, et d'écrire $(n_0,\widetilde{n_0})$ comme premier compteur, sont codées dans le graphe de jeu. On peut se convaincre assez facilement que toutes ces contraintes peuvent se coder dans un graphe de jeu de taille polynomiale.

Comme précédemment, on teste d'abord la validité des compteurs puis celles des triplets. Pour la validité des compteurs, on commence par celle des compteurs de niveau 1 (c'est $\mathcal{A}_1$ qui s'en charge), puis de niveau 2 (c'est $\mathcal{A}_2$ qui s'en charge) et ainsi de suite. Plus précisément :

- l'automate $\mathcal{A}_1$ copie le mot qu'il lit dans sa pile. S'il voit une contestation, il la vérifie comme lorsque $k = 1$. Plus précisément, il dépile N bits pour les contestations concernant la copie. Pour celles concernant la valeur, il dépile pour déterminer l'indice j du bit incriminé et, il analyse ensuite, selon la représentation contestée, les j premier bits ou les j derniers bits du compteur précédent. Si la contestation était fondée, c'est-à-dire qu'Adam avait raison, $\mathcal{A}_1$ empile infiniment le symbole ♠, sinon il empile infiniment le symbole ♡.
- l'automate $\mathcal{A}_2$ copie le mot qu'il lit dans sa pile. S'il trouve à un moment un symbole $!_c^2$ ou un symbole $\widetilde{!_c^2}$, il doit dépiler dans sa pile afin de retrouver le bit de même indice dans le compteur de niveau 2 précédent. Pour ce faire, il utilise le compteur de niveau 1 qui suit le symbole de contestation. Il procède comme l'automate $\mathcal{A}_2$ dans le cas $k = 1$. Plus précisément, il dépile jusqu'à trouver un compteur de niveau 1. Il le compare alors à la première itération du compteur de niveau 1 du mot en entrée. En cas d'égalité, il peut conclure, sinon, il dépile les itérations du compteur testé jusqu'à trouver un autre compteur de niveau 1. Il recommence alors le test. Au bout d'un moment, il trouve le bon compteur, puisqu'il possède suffisamment d'itérations pour mener à bien cette recherche exhaustive.

Dans le cas d'une contestation $!_v^2$ ou $\widetilde{!_v^2}$, appelons j l'indice du bit incriminé, qui est borné par $(tow(1,N)-1)$. Pour tester la validité de la contestation, on doit calculer la valeur que devrait prendre le bit à partir du compteur précédent. Si la contestation est $!_v^2$, il suffit de considérer les seconds éléments des paires d'indices $(tow(1,N)-1),\cdots,j$. Si la contestation est $\widetilde{!_v^2}$, il suffit de considérer les seconds éléments des paires d'indices $(tow(1,N)-1-j),\cdots,0$. Dès lors, dans ces deux cas, il faut respectivement trouver le bit d'indice j et le bit d'indice $tow(1,N)-1-j$ dans la pile de $\mathcal{A}_2$, et faire ensuite une addition d'un bit à la volée (ce qui est facile). La recherche de l'indice j se fait comme précédemment. Pour l'indice $tow(1,N)-1-j$, il suffit de remarquer que sa représentation binaire s'obtient à partir de celle de j en changeant tous les bits valant 0 en 1 et ceux valant 1 en 0. On peut donc le rechercher comme précédemment en inversant les bits.

Une fois la contestation testée, si elle était fondée, $\mathcal{A}_2$ empile infiniment le symbole ♠, sinon il empile infiniment le symbole ♡.

Enfin, si à un moment $\mathcal{A}_2$ lit un symbole ♡ ou ♠, il s'arête et empile infiniment le symbole dans sa pile. Ce cas correspond à la détection par $\mathcal{A}_1$ d'une contestation, invalide ou valide, sur un compteur de niveau 1. L'automate $\mathcal{A}_2$ ne fait alors que relayer cette information.

– pour $3 \leq h \leq k$, $\mathcal{A}_h$ fonctionne comme $\mathcal{A}_2$, sauf qu'il se concentre sur les contestations de la forme $!_c^h$, $\widetilde{!_c^h}$, $!_v^h$, $\widetilde{!_v^h}$. Une dernière différence est que $\mathcal{A}_h$ ne copie pas dans sa pile les symboles relatifs à des compteurs de niveau $h-2$. Ces derniers ne servent en effet plus à rien pour tester une contestation, et l'on peut donc les ignorer.
– $\mathcal{A}_{k+1}$ copie le mot donné en entrée, sauf qu'il ignore les compteurs de niveau $k+2$. Il va dans l'état final s'il lit ♡, et dans un état non final et bloquant s'il lit ♠. Enfin, en cas de contestation ?, il fonctionne comme $\mathcal{A}_2$ dans le cas $k=1$.

On prouve comme précédemment que l'on obtient bien une réduction polynomiale.

6.5 Ensembles de positions gagnantes

Dans ce paragraphe, on s'intéresse aux ensembles de positions gagnantes pour Eve et Adam dans un jeu sur un graphe de processus à pile muni d'une condition de la forme $\Omega^{int}_{\mathcal{A}_1\rhd\cdots\rhd\mathcal{A}_n\rhd\mathcal{A}_{n+1}}$.

Pour les conditions de la forme $\Omega^{int}_{\mathcal{A}_1\rhd\cdots\rhd\mathcal{A}_n\rhd\mathcal{A}_{n+1}}$, on ne peut utiliser les résultats du chapitre 4. En particulier, ces conditions ne sont pas invariantes par translations verticales, puisque ces dernières modifient la limite de pile.

En fait, on prouve, sans grande surprise, que les ensembles de positions gagnantes peuvent ne pas être réguliers. Plus précisément :

Proposition 6.3 *Soit un alphabet A, et soit un langage algébrique déterministe $L \subseteq A^*$. Alors, il existe un automate à pile déterministe de Büchi $\mathcal{B}$, un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$, un état $q \in Q$, et une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de Q, tels que si*

l'on appelle $\mathcal{G}$ le graphe de jeu induit par $\mathcal{P}$ et la partition précédente, l'ensemble $\{\sigma \in \Gamma^* \mid (q,\sigma)$ est gagnante pour Eve dans $(\mathcal{G},\Omega_{\mathcal{B}}^{int})\}$ est le langage algébrique L.

Preuve. Soit $\# \notin A$ un nouveau symbole. On pose $\Gamma = A \cup \{\#\}$ et $\mathcal{P} = \langle\{p,q\},\Gamma,\bot,\Delta\rangle$, où Δ est défini par : $\Delta(q,a) = \{push(p,\#)\}$ pour tout $a \in A$, et $\Delta(p,\#) = \{push(p,\#)\}$. En d'autres termes, $\mathcal{P}$ est déterministe et empile infiniment le symbole $\#$ depuis une configuration initiale de la forme (q,σ). Ainsi, dans une partie débutant en (q,σ), la pile explose strictement, et a pour limite $\sigma\#^\omega$.

L'automate $\mathcal{B}$ est choisi de telle sorte à ce que $L(\mathcal{B}) = L\#^\omega$. Il est facile de voir que $\mathcal{B}$ peut être choisi déterministe. Dès lors, une position (q,σ) est gagnante pour Eve si et seulement si $\sigma\#^\omega \in L(\mathcal{B})$, c'est-à-dire si et seulement si $\sigma \in L$. □

Une question naturelle est alors de se demander s'il y a une réciproque à la proposition 6.3, plus précisément, si les ensembles de positions gagnantes, dans un jeu sur un graphe de processus à pile équipé d'une condition de la forme $\Omega^{int}_{\mathcal{A}_1\rhd\cdots\rhd\mathcal{A}_n\rhd\mathcal{A}_{n+1}}$, sont systématiquement des langages déterministe à pile. La question reste ouverte.

6.6 Stratégies gagnantes

6.6.1 Stratégies persistantes

Dans le cas des conditions d'explosion et d'explosion stricte, nous avons vu qu'Eve possède des stratégies positionnelles. On peut se demander si c'est encore le cas pour les conditions de la forme $\Omega^{int}_{\mathcal{A}_1\rhd\cdots\rhd\mathcal{A}_n\rhd\mathcal{A}_{n+1}}$. On donne ici un résultat un peu plus faible, à savoir qu'il existe des stratégies persistantes.

Proposition 6.4 *Soit un jeu $\mathbb{G}$ sur un graphe de processus à pile équipé d'une condition de gain de la forme $\Omega^{int}_{\mathcal{A}_1\rhd\cdots\rhd\mathcal{A}_n\rhd\mathcal{A}_{n+1}}$. Depuis toute position gagnante, Eve possède une stratégie persistante.*

En fait, on prouve un résultat plus général, qui implique la proposition 6.4

Proposition 6.5 *Soit un jeu $\mathbb{G}$ sur un graphe non étiqueté muni d'une condition de gain Ω. Supposons que les deux conditions suivantes sont satisfaites :*

1. *dans toute partie $\Lambda \in \Omega$, tout sommet n'est visité qu'un nombre fini de fois.*
2. *l'ensemble Ω est fermé par sous-mots : si Λ' est un sous-mot d'une partie $\Lambda \in \Omega$, alors $\Lambda' \in \Omega$.*

Alors, depuis toute position gagnante pour Eve, celle-ci possède une stratégie gagnante assurant que tout sommet est visité au plus une fois dans une partie. En particulier, une telle stratégie est persistante.

Preuve. Notons $\mathbb{G} = (\mathcal{G},\Omega)$ et $\mathcal{G} = ((V,E),V_{\mathbf{E}},V_{\mathbf{A}})$. Soit v une position gagnante pour Eve dans $\mathbb{G}$, et soit φ une stratégie gagnante pour Eve depuis v. On appelle $\mathcal{T}_\varphi$ l'arbre V-étiqueté de stratégie associé à φ.

On commence par remarquer que la première condition implique que, pour tout sommet $w \in V$ apparaissant dans $\mathcal{T}_\varphi$, il existe un sous-arbre infini de $\mathcal{T}_\varphi$, de racine étiquetée par w et tel que w n'étiquette aucun autre sommet dans ce sous-arbre. En effet, si tel n'est pas la cas, il existerait un sous-arbre $\mathcal{T}_1$ de $\mathcal{T}_\varphi$ qui contient un

sommet étiqueté par w. Soit $\mathcal{T}_2$ un sous-arbre de $\mathcal{T}_1$ de racine étiquetée par w. De même, $\mathcal{T}_2$ possède un sous-arbre $\mathcal{T}_3$, de racine étiquetée par w. On construit donc une suite infinie décroissante d'arbres infinis $\mathcal{T}_\varphi \supsetneq \mathcal{T}_1 \supsetneq \mathcal{T}_2 \supsetneq \mathcal{T}_3 \supsetneq \cdots$ de racine étiquetée par w. L'étiquetage du chemin infini dans $\mathcal{T}_\varphi$, partant de la racine et visitant toutes les racines des sous-arbres $\mathcal{T}_i$, pour $i \geq 1$, est une partie Λ dans $\mathbb{G}$ où Eve respecte φ. Ainsi $\Lambda \in \Omega$ et visite infiniment souvent w, ce qui est contradictoire avec la première condition.

Maintenant, on peut décrire une stratégie gagnante φ' pour Eve ayant les propriétés voulues. La stratégie φ' utilise un arbre infini $\mathcal{T}$ comme mémoire. Au départ $\mathcal{T} = \mathcal{T}_\varphi$. Depuis une position $w \in V_{\mathbf{E}}$, on considère un sous-arbre $\mathcal{T}_w$ de $\mathcal{T}$ de racine w et ne contenant pas d'autre occurrence de w. Enfin, on considère l'étiquette w' du fils de la racine de $\mathcal{T}_w$. On pose $\varphi'(w,\mathcal{T}) = w'$ et la mémoire est mise à jour en posant $\mathcal{T} = \mathcal{T}_w$.

Il est facile de voir qu'une partie où Eve respecte φ' ne passe pas deux fois par un même sommet. Par ailleurs, une telle partie est un sous-mot de l'étiquetage d'une branche infinie dans $\mathcal{T}_\varphi$, c'est-à-dire que c'est un sous-mot d'un élément de Ω. Dès lors, la seconde condition nous permet de conclure que la partie est remportée par *Eve*. □

6.6.2 Stratégies effectives

On s'interroge enfin, de façon assez informelle, sur la question suivante : les méthodes présentées dans ce chapitre nous permettent-elles de proposer des stratégies gagnantes effectives pour Eve ? Si oui, quel modèle de machines peut les représenter?

Dans les preuves des théorèmes 6.4 et 6.5, nous avons montré comment construire une stratégie φ effective dans un jeu $\widetilde{\mathbb{G}}$ joué sur un graphe fini muni d'une condition de la forme $\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$ à partir d'une stratégie Φ dans un jeu $\mathbb{G}$ joué sur un graphe de processus à pile muni d'une condition de la forme $\Omega^{int}_{\mathcal{A}_2 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$. L'idée principale était de construire, à partir d'une partie partielle λ dans $\widetilde{\mathbb{G}}$, une partie partielle équivalente Λ dans $\mathbb{G}$. Ceci était fait par une application τ, qui code dans Λ le calcul de $\mathcal{A}_1$ sur l'étiquetage de λ. La valeur de $\Phi(\Lambda)$, était donnée par celle de $\varphi(\lambda)$. Supposons que la stratégie Φ utilise une pile. Dans ce cas, la stratégie φ utilise également une pile sur un alphabet plus gros (dans lequel on code un calcul de $\mathcal{A}_1$). Par exemple, comme 'on a des stratégies à pile effectives pour les jeux de parité sur un graphe de processus à pile, il en est de même pour les jeux sur des graphes finis muni d'une condition de la forme $\Omega^{ext}_{\mathcal{A}_1}$.

Considérons maintenant la transformation inverse, d'un jeu $\mathbb{G}$ sur un graphe de processus à pile vers un jeu $\widetilde{\mathbb{G}}$ sur un graphe fini. Dans la réciproque du théorème 6.6, on construit une stratégie Φ dans $\mathbb{G}$ à partir d'une stratégie φ dans $\widetilde{\mathbb{G}}$. Pour cela, on utilise une pile Π dans laquelle on stocke une partie partielle dans $\widetilde{\mathbb{G}}$. La valeur de φ sur le contenu de Π détermine le coup donné par Φ. Imaginons maintenant que la stratégie φ utilise une pile sur un alphabet S comme mémoire. On peut alors coder φ dans Π, en enrichissant l'alphabet de Π de la mémoire utilisée par φ. La nouvelle pile de stratégie a alors pour alphabet les piles sur l'alphabet S, en plus de quelques autres symboles.

En réitérant ce raisonnement, on montre que l'on peut construire des stratégies

qui utilisent comme mémoire une pile d'ordre supérieur, au sens où une pile d'ordre 1 est une pile au sens classique, et une pile d'ordre $k > 1$ est une pile dont l'alphabet de pile est composé de piles d'ordre $k - 1$. On montre qu'il existe des stratégies effectives pour Eve utilisant une pile d'ordre $n + 2$ pour des conditions de gains de la forme $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$ ou $\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_{n+1} \triangleright \mathcal{A}_{n+2}}$.

6.7 Discussion

On termine par quelques perspectives et questions ouvertes.

La collection des classes de langages $(\mathbb{C}_n(A))_{n\in\mathbb{N}}$ mériterait une étude à part. En particulier, on pourrait s'interroger sur son rapport avec celle des langages ω-algébriques. Plus précisément, existe-t-il un langage dans une classe $\mathbb{C}_n(A)$ qui ne soit pas algébrique? Si ce n'est pas le cas, on peut alors en conclure que l'inclusion de l'union des classes $\mathbb{C}_n(A)$ dans celle des langages algébrique est stricte. En effet, il existe des langages algébriques non boréliens[36], alors que tous les langages dans les classes précédentes sont boréliens. Du point de vue des jeux, une autre question intéressante concerne les propriétés de clôture par les opérations booléennes de la collection précédente. Les clôtures par union et par intersection permettraient d'utiliser les méthodes présentées dans ce chapitre pour décider si un joueur peut assurer deux propriétés, chacune décrite par un langage dans la collection précédente. La complémentation est également très intéressante. En effet, nous nous sommes attachés à décrire des stratégies pour Eve, et dès lors une procédure de complémentation nous permettrait de symétriser le problème et de donner des stratégies pour Adam.

La hiérarchie borélienne ne se limitant pas aux boréliens de rang fini, on pourrait se demander s'il existe une condition de gain borélienne de rang infini qui soit décidable. On peut se poser la même question pour une condition de gain non borélienne.

Concernant les stratégies, nous avons montré qu'il existe des stratégies gagnantes persistantes pour Eve. On peut se poser deux questions : en-est-il de même pour Adam, et peut-on étendre le résultat en montrant qu'il existe des stratégies positionnelles? Il est clair que la clôture par complémentation évoquée ci-dessus donnerait une réponse positive à la première question.

Enfin, concernant les ensembles de positions gagnantes, nous avons montré que ces derniers ne sont pas réguliers. Pour cela, nous avons exhibé une condition de la forme $\Omega^{int}_{\mathcal{B}}$ pour laquelle les positions gagnantes pour Eve formaient un langage algébrique déterministe. Il serait intéressant de caractériser exactement la forme des ensembles de positions gagnantes, voire de les représenter finiment. On pourrait par exemple se demander si l'on ne peut définir un analogue des classes $(\mathbb{C}_n(A))_{n\in\mathbb{N}}$ pour les mots finis, dont le premier niveau serait les langages algébriques déterministes. Une conjecture serait alors que l'ensemble des positions gagnantes pour Eve dans un jeu sur un graphe de processus à pile équipé d'une condition de la forme $\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$ est un ensemble dans la classe de niveau n.

Chapitre 7

Simplifications du modèle

Sommaire

Dans ce chapitre, on étudie les jeux de parité sur des graphes de Bpa et sur des graphes de processus à compteur. On peut tout d'abord s'interroger sur la pertinence d'une telle étude puisque les résultats concernant les jeux de parité sur des graphes de processus à pile s'appliquent aussi pour les cas particuliers que sont les jeux que l'on se propose d'étudier dans ce chapitre.

Concernant les Bpa, une simple remarque justifie notre étude : un jeu de parité sur un graphe de Bpa est trivial s'il est défini de la même façon que sur un graphe de processus à pile. En effet, un Bpa n'a qu'un seul état, et, dès lors tous les sommets seraient la propriété d'un même joueur et auraient tous la même couleur. On propose deux nouvelles façons de partager et de colorier les configurations d'un graphe de Bpa afin de définir un graphe de jeu et ensuite un jeu de parité. La première construction considère la lettre en sommet de pile pour attribuer la configuration à l'un des deux joueurs et pour la colorier. On parle alors de *condition locale*. La seconde construction considère l'intégralité de la pile, et l'on parle alors de *condition globale*.

On commence par l'étude des jeux sur des Bpa munis de conditions locales, et l'on montre principalement que décider le gagnant dans un tel jeu est polynomialement équivalent à décider le gagnant dans un jeu sur un graphe fini. L'étude des jeux sur des Bpa munis de conditions globales est réalisée en proposant une réduction, exponentielle, vers un jeu muni d'une condition locale. On en déduit alors que le

problème de décision du gagnant est dans DEXPTIME, et l'on montre que cette borne est également une borne inférieure.

Décider le gagnant dans un jeu d'accessibilité sur un graphe de processus à pile est un problème DEXPTIME-complet [83, 84]. La preuve (présentée dans la paragraphe 6.4.1), qui code un calcul d'une machine de Turing dans la pile, ne marche plus pour les graphes de processus à compteur. Ainsi, il était naturel de se demander si la borne en DEXPTIME pouvait être améliorée dans le cas des jeux sur des processus à compteur. On montre, en utilisant des techniques proches de celles utilisées par O. Kupferman et M. Vardi dans [51], que l'on peut résoudre le problème en PSPACE. Pour ce qui est des bornes inférieures, on montre que le problème est DP-dur.

Enfin, on mentionne deux conséquences des résultats pour les jeux sur des processus à compteur. La première est que le model-checking du μ-calcul pour les processus à compteur est un problème PSPACE qui est DP-dur. La seconde concerne les jeux sur les graphes de processus à pile munis d'une condition de gain qui est une combinaison booléenne d'une condition de parité et d'une condition sur la hauteur de pile (explosion, bornage, explosion stricte ou répétition), et fait l'objet du paragraphe 8.2 du chapitre 8.

7.1 BPA

On souhaite considérer des jeux d'accessibilité, de Büchi et de parité sur des graphes de BPA. Cependant, on ne peut utiliser les définitions données dans le cadre des jeux sur des graphes de processus à pile, car le caractère final ou la couleur d'une configuration dépend de l'état de contrôle. Dès lors, comme il n'y a pas d'état de contrôle dans le cas des BPA, le caractère final ou la couleur d'une configuration, ainsi que son appartenance à l'un des deux joueurs, doit dépendre de la pile. Nous considérons deux cas. Dans le premier, tout dépend du sommet de pile et l'on parle alors de *conditions locales*. Dans le second, plus général, tout dépend de l'appartenance du mot de pile à des langages réguliers et l'on parle alors de *conditions globales*.

7.1.1 Jeux sur des BPA munis de conditions locales

Dans tout ce paragraphe, on considère un BPA non étiqueté $\mathcal{B} = \langle \Gamma, \bot, \Delta \rangle$, muni d'une partition $\Gamma_{\mathbf{E}} \cup \Gamma_{\mathbf{A}}$ de Γ. De celle-ci découle une partition $V_{\mathbf{E}} \cup V_{\mathbf{A}}$ des sommets du graphe $G = (V,E)$ engendré par $\mathcal{B}$: les configurations appartenant à Eve sont celles dont le sommet de pile appartient à $\Gamma_{\mathbf{E}}$. On appelle $\mathcal{G}$ le graphe de jeu ainsi obtenu.

On considère enfin un ensemble fini $C \subset \mathbb{N}$ de couleurs ainsi qu'une fonction de coloriage $\rho : \Gamma \to C$ associant à toute lettre une couleur. La fonction ρ s'étend alors en une fonction de coloriage sur V en posant $\rho(v) = \rho(top(v))$, où $top(v)$ désigne le sommet de la pile dans la configuration v. Le jeu de parité $\mathbb{G}$ sur $\mathcal{G}$ est qualifié de *jeu sur un* BPA *muni d'une condition de parité locale*.

Exemple 7.1 Nous donnons un exemple qui illustrera les constructions présentées dans la suite. On considère un BPA $\mathcal{B} = \langle \Gamma, \bot, \Delta \rangle$ ainsi qu'une fonction de coloriage

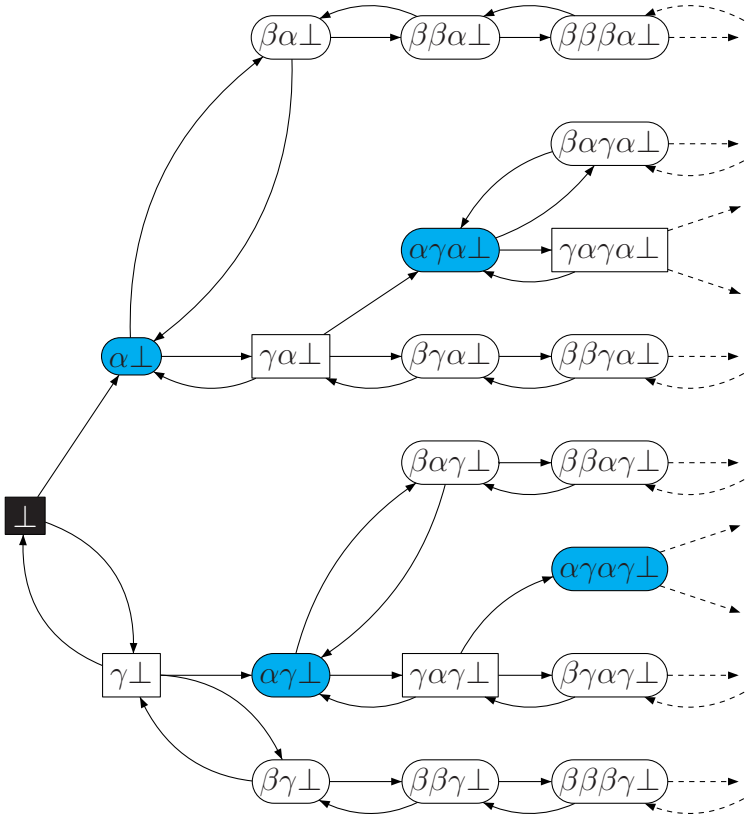


FIG. 7.1 – *Structure locale du graphe de jeu* $\mathcal{G}$

ρ définis par :

- $\Gamma = \{\alpha,\beta,\gamma,\bot\}$ avec $\Gamma_{\mathbf{E}} = \{\alpha,\beta\}$ et $\Gamma_{\mathbf{A}} = \{\gamma,\bot\}$.
- Δ est donnée par :
  - $\Delta(\alpha) = \{push(\beta),push(\gamma)\}$.
  - $\Delta(\beta) = \{pop,push(\beta)\}$.
  - $\Delta(\gamma) = \{pop,push(\alpha),push(\beta)\}$.
  - $\Delta(\bot) = \{push(\alpha),push(\gamma)\}$.
- $\rho(\alpha) = 0$, $\rho(\beta) = \rho(\gamma) = 1$ et $\rho(\bot) = 2$.

La structure locale de $\mathcal{G}$ autour de la configuration de pile vide est illustrée dans la figure 7.1, dans laquelle les sommets bleus (ou gris) sont ceux de couleur 0, les sommets noirs sont ceux de couleur 2, et les sommets blancs sont ceux de couleur 1.

Comme nous l'avions fait dans le cadre des jeux sur des graphes de processus à pile, on considère les ensembles de retours. Cependant, comme il n'y a qu'un seul

état de contrôle, il ne peut y avoir que trois cas, si l'on se restreint aux ensembles minimaux de retours : soit il n'y en a pas, soit l'ensemble vide est un ensemble de retours, soit le singleton $\{p\}$ est l'ensemble de retours minimal (si p désigne l'unique état du BPA). On définit donc les notions suivantes.

Définition 7.1 *On considère la partition $\Gamma_w \cup \Gamma_l \cup \Gamma_?$ suivante de $\Gamma \setminus \{\bot\}$:*

- *une lettre $\gamma \in \Gamma \setminus \{\bot\}$ appartient à Γ_w si et seulement si Eve possède une stratégie gagnante depuis $\gamma\bot$ dans le jeu $\mathbb{G}$ telle que la pile ne soit jamais vidée.*
- *une lettre $\gamma \in \Gamma \setminus \{\bot\}$ appartient à Γ_l si et seulement si Adam possède une stratégie gagnante depuis $\gamma\bot$ dans le jeu $\mathbb{G}$ telle que la pile ne soit jamais vidée.*
- *une lettre $\gamma \in \Gamma \setminus \{\bot\}$ appartient à $\Gamma_?$ si elle n'appartient ni à Γ_w ni à Γ_l.*

On va adapter les résultats et les techniques du chapitre 4 pour montrer que les ensembles de positions gagnantes sont réguliers. Remarquons qu'une adaptation des résultats est nécessaire, puisque la partition des configurations entre les deux joueurs est désormais faite selon le sommet de pile.

Remarque 7.1 De façon abusive, voyons $\mathcal{B}$ comme un processus à pile à un seul état p. Alors, pour toute lettre $\gamma \in \Gamma$:

- on a $\gamma \in \Gamma_w$ si et seulement si $\mathcal{R}(p,\gamma) = \{\emptyset,\{p\}\}$.
- on a $\gamma \in \Gamma_l$ si et seulement si $\mathcal{R}(p,\gamma) = \emptyset$.
- on a $\gamma \in \Gamma_?$ si et seulement si $\mathcal{R}(p,\gamma) = \{\{p\}\}$.

Dans ce cadre, l'automate alternant reconnaissant les positions gagnantes décrit dans le paragraphe 4.2.2, est désormais déterministe (si on le décrit par rapport aux ensembles de retours minimaux), et accepte exactement le langage $\Gamma_?^*\Gamma_w(\Gamma \setminus \{\bot\})^*\bot$ si $(p,\bot)$ est perdante pour Eve, et $\Gamma_?^*\Gamma_w(\Gamma \setminus \{\bot\})^*\bot \cup \Gamma_?^*\bot$ si $(p,\bot)$ est gagnante pour Eve.

L'intuition donnée dans la remarque 7.1, se vérifie et se prouve formellement.

Lemme 7.1 *Dans le jeu $\mathbb{G}$, l'ensemble des positions gagnantes pour Eve est le langage régulier décrit par l'expression rationnelle $\Gamma_?^*\Gamma_w(\Gamma \setminus \{\bot\})^*\bot$, si $\bot$ est perdante pour Eve, et par l'expression rationnelle $\Gamma_?^*\Gamma_w(\Gamma \setminus \{\bot\})^*\bot \cup \Gamma_?^*\bot$ sinon.*

Preuve. On commence par montrer que toute configuration de la forme $\gamma\sigma \in \Gamma_w(\Gamma \setminus \{\bot\})^*\bot$ est gagnante pour Eve. La preuve est la même que celle du lemme 4.1 et l'on se contente donc d'en donner l'idée : Eve joue depuis $\gamma\sigma$ comme si elle partait de $\gamma\bot$. Dès lors, la partie obtenue est un translaté vers le haut par σ d'une partie gagnante depuis $\gamma\bot$, et est donc remportée par Eve.

On considère maintenant une configuration $\sigma\gamma\sigma' \in \Gamma_?^*\Gamma_w(\Gamma \setminus \{\bot\})^*\bot$ et l'on prouve qu'elle est gagnante pour Eve dans $\mathbb{G}$. Pour cela, on procède par récurrence sur $|\sigma|$. Le résultat est vrai pour $|\sigma| = 0$ d'après ce qui précède, et on le suppose établi quand $|\sigma| = \ell$ pour un $\ell \geq 0$. Considérons donc une configuration de la forme précédente avec $|\sigma| = \ell + 1$. On a donc $\sigma = \alpha\sigma''$ avec $\alpha \in \Gamma_?$. Comme $\alpha \in \Gamma_?$, Eve possède une stratégie Φ dans $\mathbb{G}$ depuis $\alpha\bot$ telle que si la pile n'est pas vidée, elle

remporte la partie (en effet, si une telle stratégie n'existait pas, Adam posséderait une stratégie pour gagner sans vider la pile, et l'on aurait alors $\alpha \in \Gamma_l$). Dès lors, depuis $\sigma\gamma\sigma'$, Eve joue *comme* depuis $\alpha\bot$ tant que α n'est pas dépilé. Si celui-ci est un jour dépilé, on atteint alors la configuration $\sigma''\gamma\sigma'$ depuis laquelle Eve possède une stratégie gagnante (par hypothèse de récurrence), qu'elle applique. Il est alors facile de voir qu'une telle stratégie est gagnante. On prouve de même que si $\bot$ est gagnante pour Eve, il en est de même pour les configurations dans $\Gamma_?^*\bot$.

On montre symétriquement que les configurations dans $\Gamma_?^*\Gamma_l(\Gamma \setminus \{\bot\})^*\bot$ sont gagnantes pour Adam, et que si $\bot$ est gagnante pour Adam, il en est de même pour les configurations dans $\Gamma_?^*\bot$.

La preuve est donc terminée car on a :

$$(\Gamma \setminus \{\bot\})^*\bot = [\Gamma_?^*\Gamma_w(\Gamma \setminus \{\bot\})^*\bot] \cup [\Gamma_?^*\Gamma_l(\Gamma \setminus \{\bot\})^*\bot] \cup [\Gamma_?^*\bot]$$

□

Il nous faut expliquer maintenant comment calculer la partition $\Gamma_w \cup \Gamma_l \cup \Gamma_?$ de Γ. Pour cela, on construit un graphe fini $\widetilde{G} = (V,E)$, et un graphe de jeu $\widetilde{\mathcal{G}} = (\widetilde{G}, V_{\mathbf{E}}, V_{\mathbf{A}})$ qu'on équipe d'une fonction de coloriage ρ. Enfin, on appelle $\widetilde{\mathbb{G}}$ le jeu de parité sur $\widetilde{\mathcal{G}}$. Les définitions sont données par :

- $V = ((\Gamma \setminus \bot) \times \{\mathbf{E},\mathbf{A}\}) \cup \{tt, ff\}$
- les arcs sont définis comme suit: pour tout $\gamma,\gamma' \in \Gamma$ et tout $\mathbf{X} \in \{\mathbf{E},\mathbf{A}\}$.
  - il y a un arc de $(\gamma,\mathbf{E})$ vers tt si et seulement si $pop \in \Delta(\gamma)$;
  - il y a un arc de $(\gamma,\mathbf{A})$ vers ff si et seulement si $pop \in \Delta(\gamma)$;
  - il y a un arc de $(\gamma,\mathbf{X})$ vers $(\gamma',\mathbf{E})$ si et seulement si $push(\gamma') \in \Delta(\gamma)$ et $\min(\rho(\gamma),\rho(\gamma'))$ est pair;
  - il y a un arc de $(\gamma,\mathbf{X})$ vers $(\gamma',\mathbf{A})$ si et seulement si $push(\gamma') \in \Delta(\gamma)$ et $\min(\rho(\gamma),\rho(\gamma'))$ est impair.
- on a $V_{\mathbf{E}} = (\Gamma_{\mathbf{E}} \setminus \{\bot\}) \times \{\mathbf{E},\mathbf{A}\} \cup \{ff\}$.
- la fonction de coloriage ρ est étendue en une fonction sur $(\Gamma \setminus \{\bot\}) \times \{\mathbf{E},\mathbf{A}\}$ en posant $\rho((\gamma,\mathbf{X})) = \rho(\gamma)$ pour toute lettre $\gamma \in \Gamma$ et tout $\mathbf{X} \in \{\mathbf{E},\mathbf{A}\}$.

Exemple 7.2 Pour le jeu introduit dans l'exemple 7.1, on obtient le graphe de jeu $\widetilde{\mathcal{G}}$ représenté dans la figure 7.2, où les sommets bleus (ou gris) sont ceux colorés par 0 et les autres sont ceux colorés par 1.

On voit facilement que les positions gagnantes pour Eve sont $(\alpha,\mathbf{E})$, $(\alpha,\mathbf{A})$, $(\beta,\mathbf{E})$, et tt.

Donnons l'intuition qui se trouve derrière le jeu de parité $\widetilde{\mathbb{G}}$. Un sommet $(\gamma,\mathbf{E})$ sera gagnant pour Eve si et seulement si elle peut gagner depuis $\gamma\bot$ ou vider la pile. Un sommet $(\gamma,\mathbf{A})$ sera gagnant pour Eve si et seulement si elle peut gagner depuis $\gamma\bot$ sans vider la pile. En d'autres termes, la première composante γ code la configuration $\gamma\bot$, tandis que la seconde composante, $\mathbf{E}$ ou $\mathbf{A}$ spécifie à qui *profite* le vidage de la pile (Eve ou Adam).

Plus formellement, on a le résultat suivant.

Lemme 7.2 *Pour toute lettre $\gamma \in \Gamma \setminus \{\bot\}$, on a :*

- *$\gamma \in \Gamma_w$ si et seulement si $(\gamma,\mathbf{A})$ est une position gagnante pour Eve dans le jeu $\widetilde{\mathbb{G}}$.*

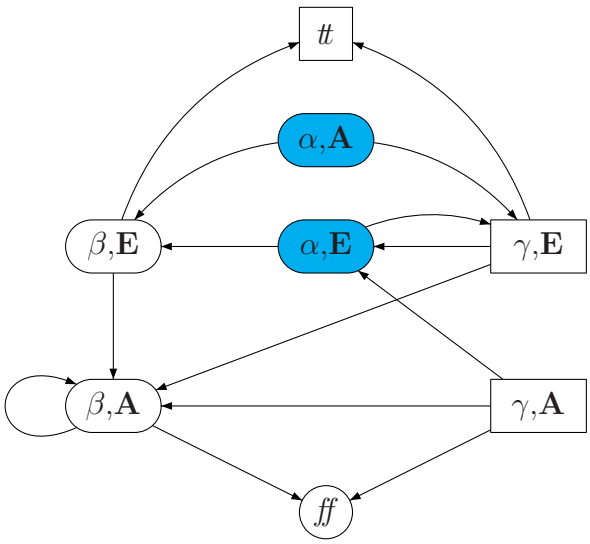


FIG. 7.2 – *Le graphe de jeu* $\widetilde{\mathcal{G}}$

– $\gamma \in \Gamma_l$ *si et seulement si* $(\gamma,\mathbf{E})$ *est une position gagnante pour Adam dans le jeu* $\widetilde{\mathbb{G}}$.

Preuve. On ne prouve que le premier point, le second s'obtenant par symétrie.

Supposons donc que $\gamma \in \Gamma_w$, c'est-à-dire qu'Eve possède une stratégie gagnante positionnelle Φ dans $\mathbb{G}$ depuis $\gamma\bot$ qui, de surcroît, assure que la pile n'est jamais vidée. A partir de Φ, on définit une stratégie gagnante, φ, pour Eve dans $\widetilde{\mathbb{G}}$ depuis $(\gamma,\mathbf{A})$. La stratégie φ utilise une mémoire infinie Λ dans laquelle on code une partie dans $\mathbb{G}$. Au départ $\Lambda = \gamma\bot$. Depuis une position $(\alpha,\mathbf{X})$ où Eve doit jouer, elle regarde la valeur de $\varphi(\Lambda)$: si c'est *pop*, elle va en *tt* (et la partie se termine), si c'est $push(\beta)$, elle va en $(\beta,\mathbf{Y})$, où $\mathbf{Y} = \mathbf{E}$ si $\min(\rho(\alpha),\rho(\beta))$ est pair et $\mathbf{Y} = \mathbf{A}$ sinon. Après chaque mouvement vers un sommet de la forme $(\beta,\mathbf{Y})$, Eve met à jour Λ en posant $\Lambda = \Lambda \cdot \beta\alpha\sigma$, où $\alpha\sigma$ était la dernière configuration de Λ. On montre alors facilement que φ est toujours définie et que l'on ne peut atteindre *ff*. En effet, si ce n'est pas le cas, on construit facilement une partie infinie dans $\mathbb{G}$, à partir de la valeur de Λ avant le coup vers *ff*, qui boucle infiniment, où Eve respecte Φ et qui est remportée par Adam. Comme la suite des couleurs dans une partie infinie λ est la même que dans la partie limite des Λ, on en déduit que λ est remportée par Eve.

Réciproquement, supposons qu'Eve possède une stratégie gagnante depuis $(\gamma,\mathbf{A})$ dans $\widetilde{\mathbb{G}}$. Considérons une telle stratégie φ, que l'on choisit de plus sans mémoire. A partir de φ, on définit une stratégie Φ dans $\mathbb{G}$ depuis $\gamma\bot$. La stratégie Φ dépend des deux symboles en sommet de pile (elle est donc positionnelle) et est définie de la façon suivante : depuis une position $\alpha\beta\sigma$, Eve considère la valeur de φ depuis $(\alpha,\mathbf{X})$, où $\mathbf{X} = \mathbf{E}$ si $\min(\rho(\alpha),\rho(\beta))$ est pair et $\mathbf{X} = \mathbf{A}$ sinon. Si le coup est vers *tt*, $\Phi(\alpha\beta\sigma) = pop$, si le coup est vers un sommet $(\eta,\mathbf{Y})$, $\Phi(\alpha\beta\sigma) = push(\eta)$.

Comme $\mathbb{G}$ est un jeu de parité, pour montrer que Φ est une stratégie gagnante depuis $\gamma\bot$, il suffit de prouver qu'elle est gagnante lorsqu'Adam joue selon une stratégie positionnelle. En particulier, on peut considérer que dès qu'une partie boucle,

elle boucle infiniment. Plus précisément, on a deux formes possibles de parties :

1. les joueurs ne font qu'empiler des symboles.
2. les joueurs empilent jusqu'à atteindre une configuration $\alpha\beta\sigma$ puis l'un d'eux dépile vers $\beta\sigma$, et l'on boucle alors en revenant vers $\alpha\beta\sigma$. La plus petite couleur infiniment souvent répétée est alors $\min(\rho(\alpha),\rho(\beta))$.

On considère donc une partie Λ depuis $\gamma\bot$ où Eve respecte Φ, et où Adam respecte une stratégie positionnelle. On voit facilement que la pile n'est pas vidée dans Λ. En effet, un tel coup ne pourrait être joué qu'au tout début de la partie. De plus, dans le cas où $\gamma \in \Gamma_{\mathbf{E}}$, on aurait, par définition de Φ, un arc de $(\gamma,\mathbf{A})$ vers tt dans $\widetilde{\mathcal{G}}$, ce qui n'est pas le cas. Lorsque $\gamma \in \Gamma_{\mathbf{A}}$, on aurait alors un arc de $(\gamma,\mathbf{A})$ vers ff dans $\widetilde{\mathcal{G}}$ et $(\gamma,\mathbf{A})$ serait gagnant pour Adam dans $\widetilde{\mathbb{G}}$. Dans le cas des parties ne vidant pas la pile, on associe à Λ une partie λ dans $\widetilde{\mathbb{G}}$ débutant en $(\gamma,\mathbf{A})$ et où Eve respecte φ. On retrouve les deux cas précédents :

1. on ne fait qu'empiler dans Λ. La partie associée λ est l'image de Λ par le morphisme lettre à lettre τ de $[(\Gamma \setminus \{\bot\})^+\bot]^\infty$ dans V^∞ où τ est défini par $\tau(\alpha\beta\sigma) = (\alpha,\mathbf{X})$, avec $\mathbf{X} = \mathbf{E}$ si $\min(\rho(\alpha),\rho(\beta))$ est pair et $\mathbf{X} = \mathbf{A}$ sinon. On voit alors immédiatement que λ est une partie valide dans $\widetilde{\mathbb{G}}$ où Eve respecte φ. De plus, les suites de couleurs visitées dans Λ et λ sont les mêmes, et comme Eve remporte λ, il en est de même pour Λ.
2. on empile au début de Λ, et ensuite on fait infiniment une boucle. La partie Λ est donc de la forme $\Lambda = \Lambda' \cdot (\beta\sigma \cdot \alpha\beta\sigma)^\omega$ avec $\alpha,\beta \in \Gamma$ et $\sigma \in \Gamma^*$, et où Λ' est une partie partielle se terminant en $\alpha\beta\sigma$ où l'on ne fait qu'empiler. Le gagnant est donc déterminé par la parité de $\min(\rho(\alpha),\rho(\beta))$. On considère à nouveau le morphisme τ précédent et l'on appelle $\lambda' = \tau(\Lambda')$. On a alors immédiatement que λ' est une partie commençant en $(\gamma,\mathbf{A})$, où Eve respecte φ, et se terminant en $(\alpha,\mathbf{X})$ où $\mathbf{X} = \mathbf{E}$ si $\min(\rho(\alpha),\rho(\beta))$ est pair et $\mathbf{X} = \mathbf{A}$ sinon. Si $\alpha \in \Gamma_{\mathbf{E}}$, on a $\Phi(\alpha\beta\sigma) = pop$ et donc $\min(\rho(\alpha),\rho(\beta))$ est pair, car, par définition de Φ, on a $\varphi(\alpha,\mathbf{X}) = tt$. Si $\alpha \in \Gamma_{\mathbf{A}}$, on a $\mathbf{X} = \mathbf{E}$, car sinon, il y aurait un arc de $(\alpha,\mathbf{X})$ vers ff dans $\widetilde{\mathcal{G}}$ et λ' pourrait être prolongée en une partie remportée par Adam alors qu'Eve respecte sa stratégie gagnante.

Dès lors, Φ est gagnante pour Eve, ce qui conclut la preuve. □

Afin de calculer l'ensemble des positions gagnantes, il nous reste à expliquer comment déterminer le joueur qui possède une stratégie gagnante depuis la configuration de pile vide $\bot$. Pour cela, on établit facilement le lemme suivant.

Lemme 7.3 *On a la caractérisation suivante :*

- *si $\bot \in \Gamma_{\mathbf{E}}$, la configuration $\bot$ est gagnante pour Eve si et seulement si l'une des deux conditions suivantes est satisfaite :*
  1. *il existe une lettre $\gamma \in \Gamma_w$ telle que $push(\gamma) \in \Delta(\bot)$.*
  2. *il existe une lettre $\gamma \in \Gamma_?$ telle que $push(\gamma) \in \Delta(\bot)$ et $\min(\rho(\bot),\rho(\gamma))$ est pair.*
- *si $\bot \in \Gamma_{\mathbf{A}}$, la configuration $\bot$ est gagnante pour Adam si et seulement si l'une des deux conditions suivantes est satisfaite :*
  1. *il existe une lettre $\gamma \in \Gamma_l$ telle que $push(\gamma) \in \Delta(\bot)$.*

2. *il existe une lettre* $\gamma \in \Gamma_?$ *telle que* $push(\gamma) \in \Delta(\bot)$ *et* $\min(\rho(\bot),\rho(\gamma))$ *est impair.*

Preuve. On se contente du premier cas, le second étant obtenu par symétrie. On suppose donc que $\bot \in \Gamma_{\mathbf{E}}$. S'il existe une lettre $\gamma \in \Gamma_w$ telle que $push(\gamma) \in \Delta(\bot)$, le résultat est direct : Eve empile γ, et applique sa stratégie gagnante depuis $\gamma\bot$. S'il existe une lettre $\gamma \in \Gamma_?$ telle que $push(\gamma) \in \Delta(\bot)$, et que de plus $\min(\rho(\bot),\rho(\gamma))$ est pair, Eve empile γ et joue depuis $\gamma\bot$, selon la stratégie qui lui permet de vider la pile ou de gagner. Si la pile n'est pas vidée, elle gagne et sinon, du fait que les stratégies des deux joueurs peuvent être choisies positionnelles, la pile est vidée juste après que l'on ait atteint $\gamma\bot$. La plus petite couleur infiniment souvent répétée est donc $\min(\rho(\bot),\rho(\gamma))$, qui est paire.

Dans les autres cas, cela signifie que pour toute règle $push(\gamma) \in \Delta(\bot)$, soit $\gamma \in \Gamma_l$, soit $\gamma \in \Gamma_?$ et $\min(\rho(\bot),\rho(\gamma))$ est impair. De la même façon, on montre que dans les deux cas, c'est Adam qui possède une stratégie gagnante. □

Nous venons de voir comment décider le gagnant depuis n'importe quelle position dans le jeu $\mathbb{G}$. Concernant la complexité du problème, on a le résultat suivant.

Théorème 7.1 *L'ensemble des positions gagnantes pour Eve dans le jeu de parité avec condition locale* $\mathbb{G}$ *peut être calculé en temps*

$$\mathcal{O}\left(dn^2\left(\frac{2n+2+d}{d}\right)^{\lceil d/2\rceil}\right)$$

où $n = |\Gamma|$ *et* $d = |C|$.

Preuve. Le résultat provient directement de la complexité du calcul des positions gagnantes dans un jeu de parité sur un graphe fini [46, 80]. □

Remarque 7.2 S'il existe un algorithme polynomial pour décider le gagnant dans un jeu de parité sur un graphe fini, il en serait de même pour calculer l'ensemble des positions gagnantes dans un jeu sur un graphe de Bpa muni d'une condition de parité locale.

Le résultat et la remarque précédente ne font que traduire le fait que le problème de décision du gagnant dans un jeu sur un Bpa muni d'une condition de parité locale se réduit polynomialement à décider le gagnant dans un jeu de parité sur un graphe fini. Réciproquement, on a facilement le résultat suivant.

Proposition 7.1 *Soit* $\widetilde{\mathbb{G}}$ *un jeu de parité sur un graphe de jeu fini* $\widetilde{\mathcal{G}}$ *et soit* v_{in} *une position dans* $\widetilde{\mathcal{G}}$*. Il existe alors un jeu* $\mathbb{G}$ *sur un* Bpa *muni d'une condition de parité locale, de même taille que* $\widetilde{\mathcal{G}}$*, tel qu'Eve possède une stratégie gagnante dans* $\widetilde{\mathbb{G}}$ *depuis* v_{in} *si et seulement si elle possède une stratégie gagnante depuis* $\bot$ *dans* $\mathbb{G}$*.*

Preuve. Notons $\widetilde{\mathcal{G}} = ((V,E),V_{\mathbf{E}},V_{\mathbf{A}})$ et appelons ρ la fonction de coloriage associée à $\widetilde{\mathbb{G}}$. On considère alors le Bpa $\mathcal{B} = \langle\Gamma,\bot,\Delta\rangle$ défini par :

- $\Gamma = V_{\mathbf{E}} \cup V_{\mathbf{A}} \cup \{\bot\}$.
- $\Delta(\bot) = \{push(v_{in})\}$.

– $\Delta(u) = \{push(v) \mid (u,v) \in E\}$, pour tout $u \in \Gamma$, $u \neq \bot$.

Enfin, on garde la même fonction de coloriage (que l'on étend en posant $\rho(\bot) = 0$) et l'on considère le jeu de parité local $\mathbb{G}$ associé. Le résultat est alors immédiat. □

Considérons enfin les stratégies gagnantes dans les jeux de parité sur des BPA munis de condition de parité locale. On se donne une stratégie φ gagnante pour Eve sur l'ensemble de ses positions gagnantes dans $\widetilde{\mathbb{G}}$. On décrit alors une stratégie positionnelle Φ ne dépendant que des deux lettres en sommet de pile. La stratégie Φ est définie comme suit :

– si $\bot \in \Gamma_{\mathbf{E}}$, $\Phi(\bot) = push(\gamma)$ pour une lettre γ telle que $push(\gamma) \in \Delta(\bot)$ et telle que, si $\bot$ est gagnante pour Eve, soit $\gamma \in \Gamma_w$ soit $\gamma \in \Gamma_?$ et $\min(\rho(\gamma),\rho(\bot))$ est pair.
– pour toute lettre $\alpha \in \Gamma_{\mathbf{E}}$, $\alpha \neq \bot$ et pour toute configuration $\alpha\beta\sigma$, $\Phi(\alpha\beta\sigma)$ est donné par :
  – si $\min(\rho(\alpha),\rho(\beta))$ est pair, $\Phi(\alpha\beta\sigma) = push(\gamma)$ si $\varphi((\alpha,\mathbf{E})) = (\gamma,\mathbf{X})$ pour $\mathbf{X} \in \{\mathbf{E},\mathbf{A}\}$, et sinon $\Phi(\alpha\beta\sigma) = pop$.
  – si $\min(\rho(\alpha),\rho(\beta))$ est impair, $\Phi(\alpha\beta\sigma) = push(\gamma)$ si $\varphi((\alpha,\mathbf{A})) = (\gamma,\mathbf{X})$ pour $\mathbf{X} \in \{\mathbf{E},\mathbf{A}\}$, et sinon $\Phi(\alpha\beta\sigma) = pop$.

La stratégie ainsi définie est gagnante pour Eve comme le montre le résultat suivant.

Proposition 7.2 *La stratégie Φ est une stratégie gagnante pour Eve dans $\mathbb{G}$ depuis toute position gagnante pour elle. De plus, la valeur de Φ dans une configuration ne dépend que des deux lettres en sommet de pile.*

Preuve. Le résultat est vrai pour les configurations de la forme $\gamma\bot$ avec $\gamma \in \Gamma_w$ d'après le lemme 7.2. Il est encore vrai pour toute position gagnante de pile non vide, en vertu de la composition des stratégies donnée par la preuve du lemme 7.1. Enfin, si la position $\bot$ est gagnante pour Eve, l'extension du résultat est immédiate. □

La proposition précédente est optimale comme le montre le résultat suivant.

Proposition 7.3 *Il existe un jeu de parité local sur un* BPA *où il n'existe pas de stratégie qui dépende uniquement du sommet de pile.*

Preuve. Il suffit pour cela de considérer le BPA $\mathcal{B} = \langle\Gamma,\bot,\Delta\rangle$, la partition de Γ et la fonction de coloriage ρ donnés par :

– $\Gamma = \{\bot,\alpha,\beta,\gamma\}$.
– $\Delta(\bot) = \{push(\alpha)\}$, $\Delta(\alpha) = \{push(\beta),pop\}$, $\Delta(\beta) = \{push(\gamma)\}$ et $\Delta(\gamma) = \{push(\alpha)\}$
– $\Gamma_{\mathbf{A}} = \emptyset$.
– $\rho(\bot) = 1$, $\rho(\alpha) = 2$, $\rho(\beta) = 1$ et $\rho(\gamma) = 3$.

Eve est donc la seule à jouer et l'on peut montrer que toutes les positions sont gagnantes pour elles. En revanche, il n'existe pas de stratégie gagnante ne dépendant que du sommet de pile. En effet, si le coup choisi quand α est en sommet de pile est de dépiler, les parties commençant en $\alpha\bot$ bouclent, et sont perdues par Eve. Si

le coup choisi est d'empiler β, les parties commençant en $\alpha\bot$ ne bouclent pas mais ont 1 comme plus petite couleur infiniment souvent répétée, et sont donc perdues par Eve. □

7.1.2 Jeux sur des BPA munis de conditions globales

Nous venons de le voir dans le paragraphe précédent, les BPA munis de conditions locales ne permettent pas des comportements complexes et sont dès lors plus proches des jeux sur des graphes finis que des jeux sur des graphes de processus à pile. On propose alors une extension naturelles des BPA, les BPA *globaux*. On définit à partir de ces derniers les jeux de parité sur des BPA globaux.

A la différence d'un BPA classique, les règles de transitions pouvant être appliquées dans un BPA global dépendent du contenu de la pile, à savoir de son appartenance à des langages réguliers. Plus précisément, on donne la définition suivante.

Définition 7.2 (BPA global) *Un* BPA global *est un quadruplet* $\mathcal{B} = \langle \Gamma,\bot,(L_1,\ldots,L_d),(\Delta_1,\ldots,\Delta_d)\rangle$. *L'alphabet* Γ *est l'alphabet de pile et* $\bot \in \Gamma$ *le symbole de fond de pile. Les langages* $L_1,\ldots,L_d$ *sont des langages réguliers (décrits par des automates déterministes) qui forment une partition de l'ensemble des configurations* $(\Gamma \setminus \{\bot\})^*\bot$. *Enfin,* $\Delta_1,\ldots,\Delta_d$ *sont des applications de* Γ *dans* $\mathcal{P}(\{pop, push(\gamma) \mid \gamma \in \Gamma \setminus \{\bot\}\})$ *telles que, pour tout* $1 \leq i \leq d$, $pop \notin \Delta_i(\bot)$.

A un BPA global $\mathcal{B} = \langle \Gamma,\bot,(L_1,\ldots,L_d),(\Delta_1,\ldots,\Delta_d)\rangle$, on associe son ensemble de configuration $(\Gamma \setminus \{\bot\})^*\bot$. Enfin, on peut passer d'une configuration $\gamma\sigma$ aux configurations suivantes, où i désigne l'unique langage L_i contenant $\gamma\sigma$:

- on peut aller en σ si $pop \in \Delta_i(\alpha)$.
- on peut aller en $\gamma'\gamma\sigma$ pour tout $\gamma' \in \Gamma$ et tel que $push(\gamma') \in \Delta_i(\alpha)$.

Enfin, on associe à un BPA global un graphe infini dont les sommets sont les configurations du graphe. Il y a un arc entre deux configurations si l'on peut passer de la première à la seconde. On parle alors de *graphe de* BPA *global*.

Afin de définir un graphe de jeu sur un graphe de BPA global, on se donne une partition de l'ensemble V des configurations en deux langages réguliers, $V_{\mathbf{E}}$ et $V_{\mathbf{A}}$. Enfin, on considère la fonction de coloriage ρ de V dans $\{1,\ldots,d\}$ définie par $\rho(\sigma) = i$, où i est l'unique entier tel que $\sigma \in L_i$. Un jeu de parité sur un tel graphe de jeu est qualifié de *jeu de parité sur un* BPA *global*. Dès lors, dans un tel jeu, l'appartenance à un joueur ainsi que la couleur d'une configuration dépend de l'appartenance de son contenu de pile à des langages réguliers. Enfin, les coups pouvant être joués dépendent du sommet de pile et de la couleur de la configuration courante.

Exemple 7.3 Considérons le BPA global $\mathcal{B} = \langle \Gamma,\bot,(L_1,L_2,L_3,L_4),(\Delta_1,\Delta_2,\Delta_3,\Delta_4)\rangle$ où l'on pose :

- $\Gamma = \{\alpha,\beta,\gamma,\bot\}$.
- $L_1 = \{\sigma \in (\Gamma \setminus \{\bot\})^*\bot \mid |\sigma|_\beta = 1 \mod 3\}$.
- $L_2 = \{\sigma \in (\Gamma \setminus \{\bot\})^*\bot \mid |\sigma|_\beta = 2 \mod 3 \wedge |v| = 0 \mod 2\}$.
- $L_3 = \{\sigma \in (\Gamma \setminus \{\bot\})^*\bot \mid |\sigma|_\beta = 2 \mod 3 \wedge |v| = 1 \mod 2\}$.

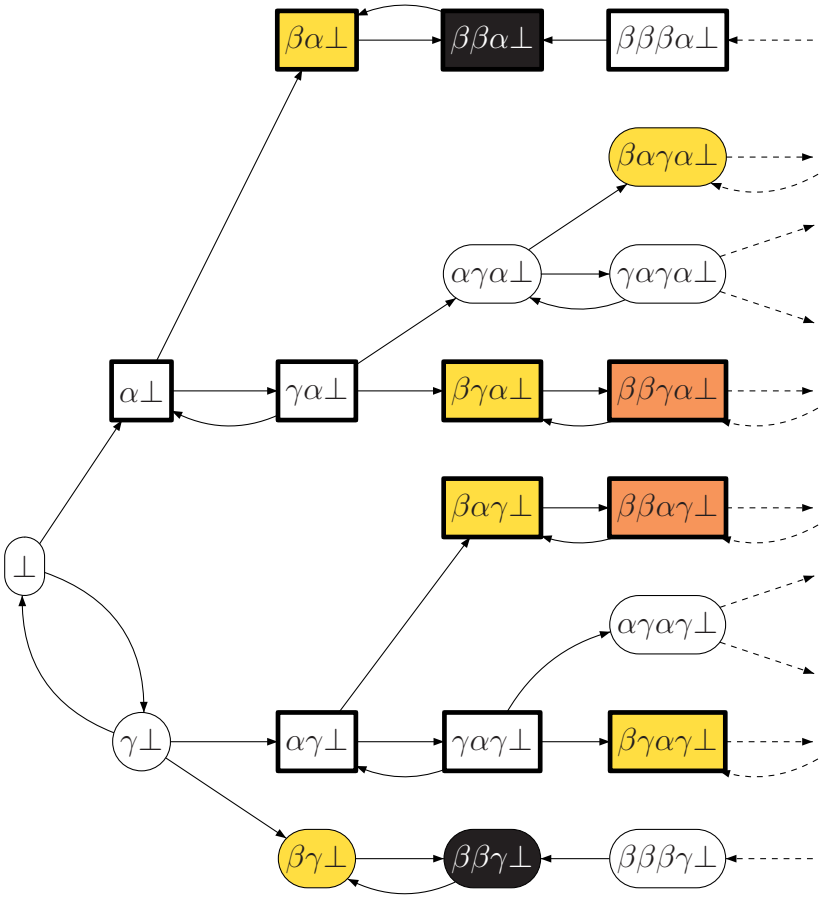


FIG. 7.3 – *Exemple de graphe de jeu de* BPA *global*

- $L_4 = \{\sigma \in (\Gamma \setminus \{\bot\})^*\bot \mid |\sigma|_\beta = 0 \mod 3\}$.
- $\Delta_1(\alpha) = \Delta_1(\gamma) = \Delta_1(\bot) = \emptyset$ et $\Delta_1(\beta) = \{push(\beta)\}$.
- $\Delta_2(\alpha) = \Delta_2(\bot) = \emptyset$, $\Delta_2(\beta) = \{pop\}$ et $\Delta_2(\gamma) = \{push(\beta)\}$.
- $\Delta_3(\alpha) = \Delta_3(\gamma) = \Delta_3(\bot) = \emptyset$ et $\Delta_3(\beta) = \{push(\beta),pop\}$.
- $\Delta_4(\alpha) = \{push(\beta),push(\gamma)\}$, $\Delta_4(\beta) = \{pop\}$, $\Delta_4(\gamma) = \{push(\alpha),push(\beta),pop\}$ et $\Delta_4(\bot) = \{push(\alpha),push(\gamma)\}$.

Le graphe de jeu induit par $\mathcal{B}$ et la partition $V_{\mathbf{E}} = \{\sigma \in (\Gamma \setminus \{\bot\})^*\bot \mid |\sigma|_\alpha = 0 \mod 2\}$ est représenté dans la figure 7.3. Les sommets de couleur 1 sont jaunes (gris clair), ceux de couleur 2 sont noirs, ceux de couleur 3 sont roses (gris foncé) et ceux de couleur 4 sont blancs.

Dans ce qui suit, on se donne un BPA global $\mathcal{B} = \langle\Gamma,\bot,(L_1,\ldots,L_d),(\Delta_1,\ldots,\Delta_d)\rangle$, une partition $V_{\mathbf{E}} \cup V_{\mathbf{A}}$ des configurations de $\mathcal{B}$. On note $\mathcal{G}$ et $\mathbb{G}$ le graphe de jeu et le jeu de parité associés.

Afin de décider qui gagne dans un tel jeu, on construit un jeu de parité local sur un BPA *équivalent*. Pour cela, on considère, pour tout $1 \leq i \leq d$, le langage régulier $L'_i = \{\sigma \in (\Gamma \setminus \{\bot\})^* \mid \sigma\bot \in L_i\}$, et le langage miroir $\widetilde{L'_i}$ de L'_i défini par $\widetilde{L'_i} = \{\sigma \in (\Gamma \setminus \{\bot\})^* \mid \widetilde{\sigma} \in L'_i\}$. Enfin, on considère pour tout $1 \leq i \leq d$, un automate déterministe $\mathcal{A}_i = \langle Q_i,\Gamma \setminus \{\bot\},q_i^{in},\delta_i,F_i\rangle$ acceptant le langage $\widetilde{L'_i}$.

De même, on se donne un automate déterministe $\mathcal{A}_E = \langle Q_E ve,\Gamma\setminus\{\bot\},q_E^{in},\delta_E,F_E\rangle$ acceptant le langage régulier $\widetilde{V'_{\mathbf{E}}} = \{\sigma \in (\Gamma \setminus \{\bot\})^* \mid \widetilde{\sigma}\bot \in V_{\mathbf{E}}\}$.

A la différence de ce que l'on faisait pour les enrichissements déterministes (voir chapitre 4), toute l'information doit être codée dans la pile. Pour cela, on définit un BPA $\mathcal{B}' = \langle \Gamma', \perp', \Delta' \rangle$ où l'on pose :

- $\Gamma' = \Gamma \times Q_E \times Q_1 \times \cdots \times Q_d$.
- $\perp'$ est identifié avec $(\perp, q_E^{in}, q_1^{in}, \ldots, , q_d^{in})$.
- pour tout $1 \leq i \leq d$, pour tout $q_i \in F_i$, pour tout $\gamma \in \Gamma$, pour tout $q_E \in Q_E$ et pour tout $q_j \in Q_j \setminus F_j$, $j \neq i$, $push((\gamma', \delta_E(q_E, \gamma'), \delta_1(q_1, \gamma'), \ldots, \delta_d(q_d, \gamma'))) \in \Delta'((\gamma, q_E, q_1, \ldots, q_d))$ si et seulement si $push(\gamma') \in \Delta_i(\gamma)$.
- pour tout $1 \leq i \leq d$, pour tout $q_i \in F_i$, pour tout $\gamma \in \Gamma$, pour tout $q_E \in Q_E$ et pour tout $q_j \in Q_j \setminus F_j$, $j \neq i$, $pop \in \Delta'((\gamma, q_E, q_1, \ldots, q_d))$ si et seulement si $pop \in \Delta_i(\gamma)$.

On partitionne alors Γ' en $\Gamma'_{\mathbf{E}} \cup \Gamma'_{\mathbf{A}}$ en posant $\Gamma'_{\mathbf{E}} = \Gamma \times F_E \times Q_1 \times \cdots \times Q_d$, et l'on définit une fonction de coloriage ρ de Γ' dans $\{1, \ldots, d\}$ en définissant $\rho((\gamma, q_E, q_1, \ldots, q_d)) = i$ si et seulement si i est l'unique indice tel que $q_i \in F_i$. Si un tel i n'existe pas ou s'il n'est pas unique, la lettre a n'importe quelle couleur (en fait une telle lettre ne pourra apparaître dans la pile dans les parties que l'on va considérer par la suite).

On appelle enfin $\mathcal{G}'$ et $\mathbb{G}'$ le graphe de jeu et le jeu de parité local engendrés par $\mathcal{B}'$, par la partition précédente, et par la fonction de coloriage ρ. Enfin, on définit la fonction τ de $(\Gamma \setminus \{\perp\})^* \perp$ dans $\Gamma'^* \perp'$ définie par $\tau(\gamma_1 \cdots \gamma_k \perp) = \gamma'_1 \cdots \gamma'_k \perp'$ où l'on pose $\gamma'_k = (\gamma_k, \delta_E(q_E^{in}, \gamma_k), \delta_1(q_1^{in}, \gamma_k), \ldots, \delta_d(q_d^{in}, \gamma_k))$, et pour tout $2 \leq i \leq k$, $\gamma'_{i-1} = (\gamma_{i-1}, \delta_E(q_E, \gamma_{i-1}) \delta_1(q_1, \gamma_{i-1}), \ldots, \delta_d(q_d, \gamma_{i-1}))$, où l'on pose $\gamma'_i = (\gamma_i, q_E, q_1, \ldots, q_d)$.

On a alors le résultat suivant.

Lemme 7.4 *Une configuration σ est gagnante pour Eve dans le jeu de parité global $\mathbb{G}$ si et seulement si $\tau(u)$ est gagnante pour Eve dans le jeu de parité local $\mathbb{G}'$.*

Preuve. La preuve est à rapprocher de celle donnée dans le paragraphe 4.3.3 pour les conditions régulières de pile, sauf que cette fois-ci tout l'information est codée dans la pile. On conclut en utilisant le fait que les stratégies dans les jeux $\mathbb{G}$ et $\mathbb{G}'$ se transposent directement de l'un vers l'autre. □

Remarque 7.3 Comme la fonction τ est rationnelle (c'est-à-dire réalisable par un transducteur), et que l'ensemble des positions gagnantes dans $\mathbb{G}'$ est régulier, on déduit facilement que l'ensemble des positions gagnantes est régulier dans un jeu de parité sur un BPA global.

Alors que dans le cas des jeux de parité locaux sur des BPA, les ensembles de postions gagnantes étaient des ensembles réguliers très simples, il est facile de voir que l'on peut obtenir n'importe quel langage régulier comme ensemble de positions gagnantes dans les jeux de parité sur un BPA global. Pour cela, si l'on se donne un ensemble régulier L, il suffit de considérer le BPA global $\mathcal{B} = \langle \Gamma, \perp, (\Gamma \setminus \{\perp\})^* \perp, \Delta \rangle$ où Δ associe à toute lettre l'ensemble vide. Enfin, si l'on prend pour $V_{\mathbf{E}} = \overline{L}$ et $V_{\mathbf{A}} = L$, le graphe de jeu obtenu ne contient pas d'arc et les positions gagnantes pour Eve sont exactement les positions où Adam doit jouer, c'est-à-dire le langage L.

Le lemme 7.4 montre que l'on peut décider le gagnant dans un jeu de parité sur un BPA global. Si l'on s'intéresse à la complexité de ce problème, on peut remarquer tout d'abord que les automates $\mathcal{A}_E$ et $\mathcal{A}_1, \ldots, \mathcal{A}_d$ peuvent être choisis de taille exponentielle (et calculés en temps exponentiel également), à partir de la description du BPA global $\mathcal{B}$ (il s'agit du calcul simple de l'automate miroir : on retourne les flèches et l'on déterminise). Dès lors, la taille de Γ' est telle que $|\Gamma'| = \mathcal{O}(2^{|\mathcal{B}|(d+1)})$, où $|\mathcal{B}|$ est la taille de la représentation de $\mathcal{B}$. On peut alors décider en temps exponentiel, en la taille de $\mathcal{B}$, si une position donnée σ est gagnante pour Eve dans le jeu $\mathbb{G}$. En effet, on commence par calculer $\tau(\sigma)$, ce qui prend un temps linéaire en $|\sigma|$, puis on calcule l'ensemble des positions gagnantes dans $\mathbb{G}'$. Ce dernier point requiert un temps exponentiel dans le nombre de couleurs d, et polynomial dans la taille de $\mathcal{B}'$. Dès lors, comme $\mathcal{B}'$ est exponentiel dans la taille de $\mathcal{B}$, l'algorithme total est exponentiel à la fois dans la taille de $\mathcal{B}$ et dans le nombre d de couleurs. On vient donc d'établir le résultat suivant.

Proposition 7.4 *Décider le gagnant dans un jeu de parité sur un* BPA *global est un problème* DEXPTIME.

Concernant la borne inférieure, on a le résultat suivant pour les conditions d'accessibilité, et donc aussi pour les conditions de parité

Proposition 7.5 *Décider le gagnant dans un jeu d'accessibilité sur un* BPA *global est un problème* DEXPTIME*-dur.*

Preuve. Comme dans [83, 84], on simule une machine de Turing alternante d'espace linéaire à l'aide d'un jeu d'accessibilité sur un BPA global. Appelons M une telle machine. Pour tout $n \geq 1$, sur une entrée de taille $n-1$, une telle machine utilise donc $n-1$ cases. On appelle Q les états de M. Ces derniers sont partitionnés en états existentiel, $Q_\exists$, et universels, $Q_\forall$. On note A l'alphabet de M. Sans perte de généralité, on peut supposer que la fonction de transition de la machine M est une fonction de $Q \times A$ dans $(Q \times A \times \{\leftarrow\ ,\ \rightarrow\})^2$: il y a toujours deux transitions possibles et la tête de lecture ne peut rester sur place. Comme à l'habitude, le mot wqw' représentera la configuration où l'état de contrôle est q, où le ruban contient le mot ww', et où la tête de lecture pointe sur la première lettre de w' (en particulier $w' \neq \varepsilon$).

Considérons une configuration initiale C, de longueur n, de la machine de Turing. On construit alors un jeu d'accessibilité $\mathbb{G}$ sur un BPA global $\mathcal{B}$, tel que M accepte depuis C si et seulement si Eve possède une stratégie gagnante depuis $\bot$ dans $\mathbb{G}$.

On appelle $D = Q \times A \times \{\leftarrow\ ,\ \rightarrow\}$ l'ensemble des transitions de M, et l'on choisit $\Gamma = A \cup Q \cup \{\#\} \cup D \cup \{\bot\}$ pour alphabet de pile de $\mathcal{B}$. La fonction de transition Δ de $\mathcal{B}$ n'autorise que des empilements et fonctionne comme suit :

1. les n premières étapes sont consacrées à empiler la description de C. Cela est fait en testant la hauteur de la pile.
2. si l'état de M dans la dernière configuration empilée est existentiel, Eve empile un élément de $d \in D$ pris dans $\delta(q,a)$ où q et a désignent respectivement l'état et la lettre sous la tête de lecture dans la dernière configuration. Sinon, c'est

Adam qui empile une telle transition. Si l'état alors atteint est final, la partie se termine et Eve gagne.

3. pour des raisons techniques, on empile alors des symboles # de sorte à obtenir une configuration de hauteur de pile congrue à 0 modulo $2n$.
4. on doit maintenant décrire la nouvelle configuration de M résultant de l'application de la transition d. Pour cela, Adam recopie l'ancienne configuration excepté aux positions critiques, celles autour de la tête de lecture, où l'on doit soigneusement mettre à jour la configuration. Cette mise à jour est contrainte et rendue valide à l'aide de langages réguliers.
5. on revient alors dans la deuxième étape.

Les différentes étapes de la description précédente seront détectées à l'aide des couleurs induites par l'appartenance du contenu de pile à des langages réguliers. Comme la condition de gain est une condition d'accessibilité, la signification *numérique* des couleurs est sans importance, et l'on se permettra d'utiliser des couleurs non associées à des entiers. La couleur d'une configuration σ de $\mathcal{B}$, ainsi que ses successeurs possibles, sont décrits par les langages suivants :

- les configurations d'Eve sont décrites par le langage régulier $V_{\mathbf{E}}$ donné par $V_{\mathbf{E}} = (\bigcup_{i=0}^{n-1} A^{n-i-1} Q_{\mathbf{E}} A^i)\Gamma^*\bot$: on vient de décrire une configuration existentielle de M.
- on introduit n couleurs : 0,1,...,$n-1$ pour l'étape d'initialisation. La couleur j, $0 \leq j \leq n-1$ est associée au langage $(A \cup Q)^j \bot$ (on a écrit j symboles). Le seul coup possible depuis une configuration de couleur j est $push(c_{j+1})$, où l'on pose $C = c_1 \dots c_n$.
- on introduit alors un ensemble de couleurs $choose(q,a)$ pour $q \in Q \setminus F$ et $a \in A$. Le langage associé à $choose(q,a)$ est $A^*aq\Gamma^* \cap Mod_n$ où $Mod_n = \bigcup_{k=n \mod 2n} \Gamma^k \bot$. Depuis une configuration de couleur $choose(q,a)$ il y a deux coups possibles : $push(d_1)$ et $push(d_2)$ où $\delta(q,a) = \{d_1,d_2\}$. Par définition de $V_{\mathbf{E}}$, le choix de la règle est effectuée par Eve si et seulement si $q \in Q_\exists$.
- pour des raisons techniques, après l'empilement d'un élément de D, on empile des symboles # jusqu'à revenir à une configuration de hauteur de pile (sans compter $\bot$) multiple de $2n$. Pour cela, on introduit la couleur $fill$ associée au langage $\bigcup_{n<j<2n} Mod_j$ où $Mod_j = \bigcup_{k=j \mod 2n,\ k>n} \Gamma^k \bot$. Depuis une configuration de couleur $fill$, le seul coup possible est $push(\#)$.
- une fois un coup choisi et la série de # empilée, on doit décrire la nouvelle configuration. Pour ce faire, on crée une couleur $write(a)$ pour toute lettre $a \in A$ et une couleur $write(q)$ pour tout état $q \in Q$. Depuis une configuration de couleur $write(a)$, on ne peut appliquer que la règle $push(a)$ et depuis une configuration de couleur $write(q)$, on ne peut appliquer que la règle $push(q)$. Pour être colorée par $push(a)$, une configuration σ doit être dans un langage $\Gamma^k \bot$ avec $k = j \mod 2n$ avec $0 \leq j < n$: on doit être dans la phase de mise à jour. On voit alors l'importance des #. La configuration σ doit également vérifier l'une des six conditions suivantes, illustrées dans la figure 7.1.2, portant sur les $2n$ derniers symboles de σ, et donc exprimables par des langages réguliers :

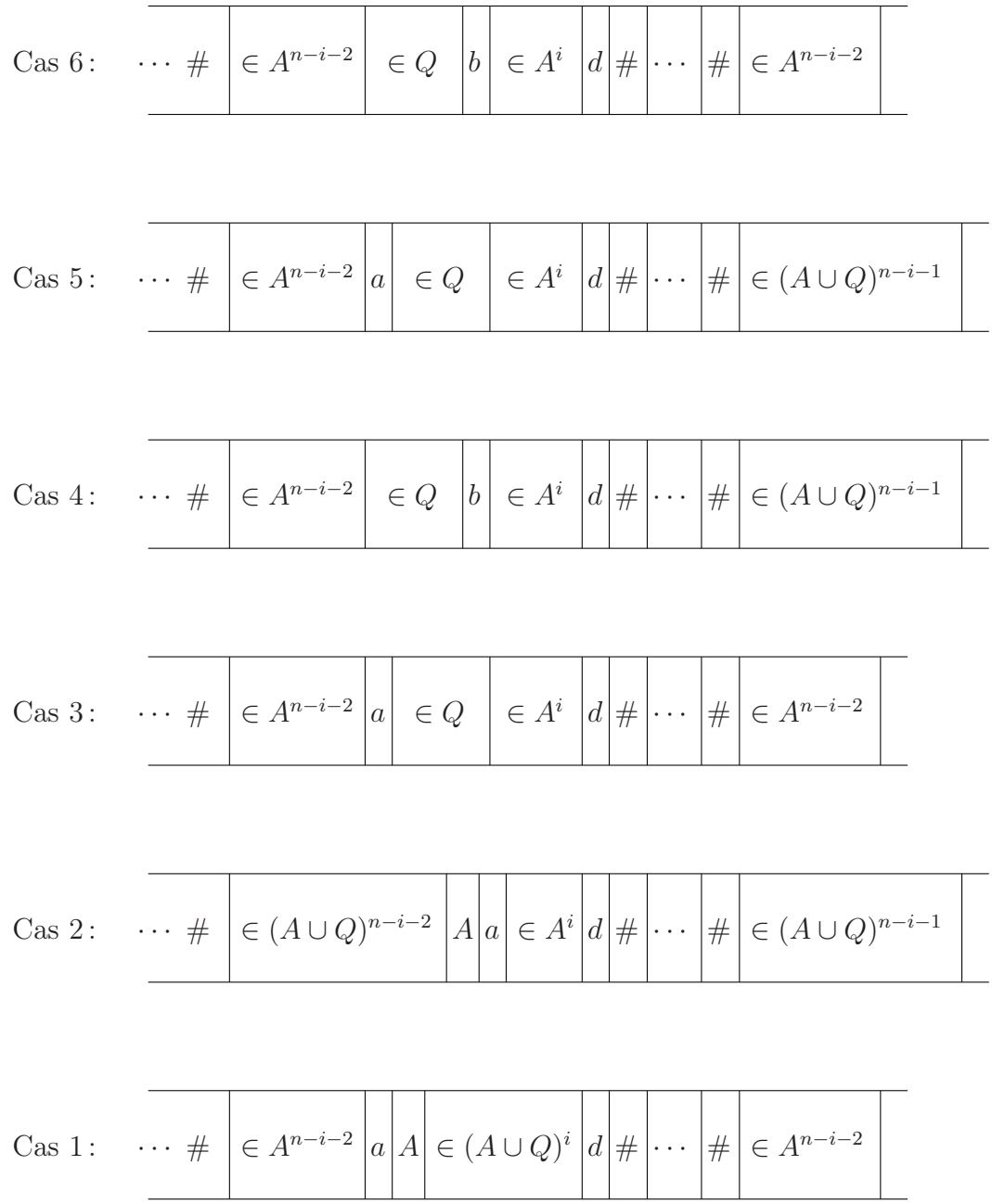


FIG. 7.4 – *Configurations de couleur write(a)*

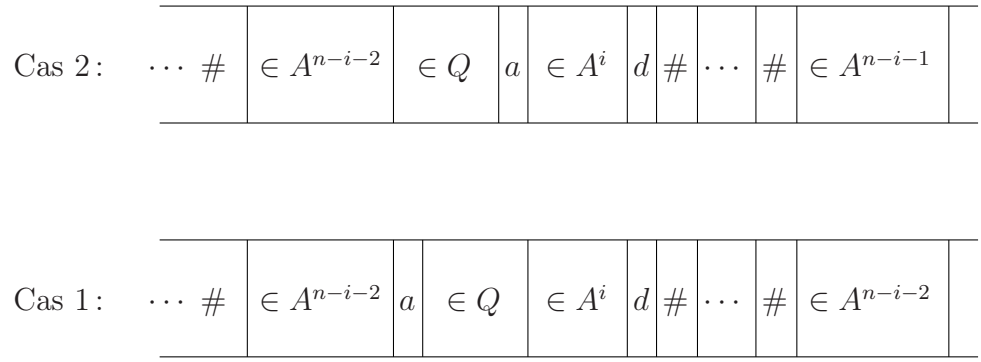


FIG. 7.5 – *Configurations de couleur write*(q)

1. pour un $0 \leq i < n$, $\sigma \in A^{n-i-2}\#^{n-1}D(A \cup Q)^i Aa\Gamma^*\perp$: a est dans la première case qui n'a pas été mise à jour. Cette dernière est suffisamment éloignée de la tête de lecture pour ne pas être modifiée.
2. pour un $0 \leq i < n$, $\sigma \in (A \cup Q)^{n-i-1}\#^{n-1}DA^iaA\Gamma^*\perp$: c'est la cas symétrique (cette fois la case est à droite de la tête de lecture).
3. pour un $0 \leq i < n$, $\sigma \in A^{n-i-2}\#^{n-1}dA^iQa\Gamma^*\perp$, pour une transition $d \in D$ de la forme $(q,b, \rightarrow)$: a est la lettre à gauche de la tête de lecture. Comme celle-ci va à droite, on doit encore avoir a.
4. pour un $0 \leq i < n$, $\sigma \in (A \cup Q)^{n-i-1}\#^{n-1}dA^ibQ\Gamma^*\perp$, pour une lettre $b \in A$, et une transition $d \in D$ de la forme $(q,a, \leftarrow)$: la lettre sous la tête de lecture était b et est remplacée par a et comme la tête va vers la gauche. La position dans la description de la nouvelle configuration est inchangée.
5. pour un $0 \leq i < n$, $\sigma \in (A \cup Q)^{n-i-1}\#^{n-1}dA^iQa\Gamma^*\perp$ pour une transition $d \in D$ de la forme $(q,b, \leftarrow)$. Dans ce cas la tête de lecture va vers la gauche et va pointer sur a, qui devient la lettre d'indice j dans la description de la nouvelle configuration, où j est l'indice de q dans l'ancienne description.
6. pour un $0 \leq i < n$, $\sigma \in A^{n-i-2}\#^{n-1}dA^ibQ\Gamma^*\perp$, pour une lettre $b \in A$, et pour transition $d \in D$ de la forme $(q,a, \rightarrow)$: la tête de lecture se déplace vers la droite et l'ancienne lettre pointée, b, est réécrite en a. La lettre a occupe dans la nouvelle description l'ancienne place de l'état de contrôle.

Enfin, pour être coloré par $write(q)$ pour $q \in Q$, une configuration σ doit être dans un langage $\Gamma^k\perp$ avec $k = j \mod 2n$ avec $0 \leq j < n$. La configuration σ doit également vérifier l'une des deux conditions symétriques suivantes, illustrées dans la figure 7.1.2, portant sur les $2n$ derniers symboles de σ :

1. pour un $0 \leq i < n$, $\sigma \in A^{n-i-2}\#^{n-1}dA^iQa\Gamma^*\perp$, pour une lettre $a \in A$, et pour une transition $d \in D$ de la forme $(q,b, \leftarrow)$: le nouvel état devient q et l'indice de la tête de lecture dans la description de la nouvelle configuration décroît d'une unité (on vient de bouger vers la gauche).
2. pour un $0 \leq i < n$, $\sigma \in A^{n-i-1}\#^{n-1}dA^iaQ\Gamma^*\perp$, pour une lettre a et pour une transition $d \in D$ de la forme $(q,a, \rightarrow)$: le nouvel état devient q, et l'in-

dice de la tête de lecture dans la description de la nouvelle configuration croît d'une unité (on vient de bouger vers la droite).

– enfin, on a une couleur marquant les configurations finales qui est décrite par le langage $D_f\Gamma^* \cup FA^*\bot$, où D_f est l'ensemble des transitions de la forme $(q,\_,\_)$ avec $q \in F$. Le premier ensemble correspond donc à l'application d'une transition conduisant la machine M dans une configuration finale, tandis que le second ensemble couvre le cas particulier où la configuration initiale est finale.

On voit facilement que tout sommet est au plus coloré par une couleur. On peut ajouter une nouvelle couleur pour les sommets non colorés (même si ceux-ci ne peuvent être atteints).

Par construction, on vérifie qu'Eve possède dans le jeu d'accessibilité associé à $\mathcal{B}$ une stratégie gagnante depuis $\bot$ si et seulement si M accepte depuis la configuration initiale C.

Enfin, pour ce qui est du caractère polynomial de la réduction, il est facile de voir que l'alphabet Γ est polynomial en la taille de M, et que les divers langages intervenant sont facilement descriptibles par des automates déterministes de taille polynomiale en n, d'où le résultat. □

7.2 Processus à compteur

On considère maintenant un jeu de parité associé à un processus à compteur et l'on souhaite décider si une configuration de pile vide donnée est gagnante pour Eve dans ce jeu. Pour cela, on utilise une technique inspirée de travaux plus généraux, dus à M. Vardi et O. Kupferman [79, 50], que nous allons brièvement et informellement présenter dans les préliminaires. On adapte ensuite cette méthode au cas des jeux sur des processus à compteur et on donne des preuves détaillées. On en déduit alors une borne supérieure (PSPACE) à notre problème. On termine enfin en donnant une borne inférieure (DP-dur).

7.2.1 Préliminaires

On considère un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$, d'alphabet $\Gamma = \{\bot,\gamma_1,\gamma_2,\ldots,\gamma_k\}$ et un jeu $\mathbb{G} = (\mathcal{G},\Omega)$ associé. On appelle $\mathcal{T}$ l'arbre infini complet d'arité k. On a alors un étiquetage naturel de $\mathcal{T}$ par les mots de pile dans $\mathcal{P}$, à savoir $(\Gamma \setminus \{\bot\})^*\bot$: la racine est étiquetée par $\bot$, et tout sommet d'étiquette σ a k fils, respectivement étiquetés par $\gamma_1\sigma,\ldots,\gamma_k\sigma$. Dès lors une partie commençant dans une position de pile vide $(p_{in},\bot)$ peut être vu comme un parcours dans l'arbre $\mathcal{T}$ par une machine $\mathcal{A}$ avec un contrôle fini dont nous précisons la nature. L'ensemble des états de contrôles est Q et dans un sommet x d'étiquette $\gamma\sigma$, lorsque la machine est dans un état q, elle peut effectuer les opérations suivantes :

– rester sur place et changer son état en q' si et seulement si $skip(q') \in \Delta(q,\gamma)$.
– remonter vers le père de x et changer son état en q' si et seulement si $pop(q') \in \Delta(q,\gamma)$.

– aller vers le fils de x d'étiquette $\gamma'\gamma\sigma$ et changer son état en q' si et seulement si $push(q',\gamma') \in \Delta(q,\gamma)$.

Enfin, pour rendre compte du caractère alternant des jeux, la machine est alternante : si q est un état d'Eve, elle effectue une des transitions possibles (coup existentiel) et si q est un état d'Adam, elle mène en parallèle toutes les opérations possibles. Ainsi, un calcul de la machine peut être vu comme un arbre infini Q-étiqueté, dans lequel un sommet étiqueté par un état d'Eve possède au plus un fils, alors qu'un sommet étiqueté par un état d'Adam peut posséder plusieurs fils. Enfin, la machine accepte s'il existe un arbre de calcul dans lequel toutes les branches infinies sont telles que la plus petite couleur (colorant les états des étiquettes) apparaissant infiniment souvent dans la branche soit paire.

La machine $\mathcal{A}$ que l'on vient de décrire informellement est un *automate d'arbres bidirectionnel alternant de parité*. On a facilement le résultat suivant.

Proposition 7.6 *Eve possède une stratégie gagnante dans $\mathbb{G}$ depuis $(p_{in},\bot)$ si et seulement si $\mathcal{A}$ accepte $\mathcal{T}$ depuis l'état initial p_{in}.*

On réduit ensuite le problème de l'acceptation de l'arbre $\mathcal{T}$ par $\mathcal{A}$ à décider le vide pour un automate d'arbre bidirectionnel alternant de parité de même taille que $\mathcal{A}$. On a enfin le résultat suivant dû à M. Vardi, permettant de déduire que l'on peut décider le gagnant dans un jeu sur un graphe de processus à pile en temps exponentiel.

Théorème 7.2 *[79] Décider le vide pour un automate d'arbre bidirectionnel alternant de parité est un problème* Dexptime*-complet.*

Pour une présentation plus détaillée de ces résultats et variantes, on consultera [79, 50, 17].

7.2.2 Borne supérieure

On considère maintenant un processus à compteur $\mathcal{C} = \langle Q,\{1,\bot\},\bot,\Delta\rangle$ muni d'une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de ses états entre Eve et Adam. On suppose que l'on a une fonction de coloriage $\rho : Q \to \{1,\ldots,d\}$. On appelle $\mathcal{G}$ le graphe de jeu associé, et $\mathbb{G}$ le jeu de parité sur $\mathcal{G}$. On souhaite décider si une configuration de pile vide $(q_{in},\bot)$ est gagnante pour Eve dans $\mathbb{G}$.

Dans le cas des processus à pile, on voyait la partie comme un parcours dans l'arbre complet d'arité $k-1$ où k est la cardinalité de l'alphabet de pile. Ainsi, dans le cas d'un processus à compteur, $k = 2$, et l'on parcours donc un mot infini. L'automate *équivalent au jeu* dont on souhaitait tester le vide est un automate bidirectionnel alternant de parité sur les mots, et non plus sur les arbres. Dans un premier temps, on fixe les notations et définitions, et dans un second temps, on montre que tester le vide pour un tel automate peut être réalisé en Pspace.

Un automate bidirectionnel alternant de parité est une machine de Turing alternante sans ruban de travail, qui ne peut écrire et qui est munie d'une condition de parité.

Définition 7.3 *Un* automate bidirectionnel alternant de parité $\mathcal{A}$ *est un quintuplet*

$\langle Q,A,\ q_{in},\delta,\rho\rangle$, où Q est un ensemble fini d'états, A est un alphabet fini d'entrée, $q_{in} \in Q$ est l'état initial, δ est une application de $Q \times A$ dans $\mathcal{B}^+(Q \times \{-1,0,1\})$ appelée fonction de transition, et ρ est une fonction de Q dans un ensemble fini de couleurs $C \subset \mathbb{N}$, appelée fonction de coloriage.

Un calcul sur un mot infini $u = a_0a_1\cdots$ est un arbre infini $(Q \times \mathbb{N})$-étiqueté, de racine $(q_{in},0)$, et tel que pour tout sommet x d'étiquette (q,n), l'ensemble des étiquettes $\{(q_1,n_1),\ldots,(q_k,n_k)\}$ des fils de x sont telles que $\{(q_1,n_1-n),\ldots,(q_k,n_k-n)\} \models \delta(q,a_n)$. Un calcul est acceptant si toutes les branches infinies satisfont la condition de parité (la couleur d'un sommet étant la couleur de l'état qui l'étiquette). Enfin, un mot infini est accepté par $\mathcal{A}$ si et seulement s'il existe un calcul acceptant de $\mathcal{A}$ dessus.

On définit l'automate bidirectionnel alternant de parité $\mathcal{A} = \langle Q,\{1,\bot\},q_{in},\delta,\rho\rangle$, où la fonction δ est donnée :

- pour tout $q \in Q_{\mathbf{E}}$, $\delta(q,1) = [\bigvee_{push(q')\in\Delta(q,1)}(q',1)] \vee [\bigvee_{skip(q')\in\Delta(q,1)}(q',0)] \vee [\bigvee_{pop(q')\in\Delta(q,1)}(q',-1)]$
  et $\delta(q,\bot) = [\bigvee_{push(q')\in\Delta(q,\bot)}(q',1)] \vee [\bigvee_{skip(q')\in\Delta(q,\bot)}(q',0)]$.
- pour tout $q \in Q_{\mathbf{A}}$, $\delta(q,1) = [\bigwedge_{push(q')\in\Delta(q,1)}(q',1)] \wedge [\bigwedge_{skip(q')\in\Delta(q,1)}(q',0)] \wedge [\bigwedge_{pop(q')\in\Delta(q,1)}(q',-1)]$
  et $\delta(q,\bot) = [\bigwedge_{push(q')\in\Delta(q,\bot)}(q',1)] \wedge [\bigwedge_{skip(q')\in\Delta(q,\bot)}(q',0)]$.

On a alors facilement le résultat suivant, qui est à rapprocher de la proposition 7.6

Proposition 7.7 *Eve possède une stratégie gagnante dans $\mathbb{G}$ depuis $(p_{in},\bot)$ si et seulement si $\mathcal{A}$ accepte le mot infini $\bot 1^\omega$.*

Il est alors aisé de modifier l'automate $\mathcal{A}$ de telle sorte que le langage reconnu par le nouvel automate (qui est de même taille que $\mathcal{A}$) soit non vide si et seulement si $\mathcal{A}$ accepte $\bot 1^\omega$. Pour cela, on considère l'automate $\mathcal{A}' = \langle Q \cup \{q_1\},\{1,\bot\},q_{in},\delta',\rho'\rangle$ défini comme suit. L'état $q_1 \notin Q$ est un nouvel état qui va permettre de vérifier que le mot en entrée est bien $\bot 1^\omega$. La fonction de coloriage ρ' étend ρ: $\rho'(q) = \rho(q)$ pour tout $q \in Q$ et $\rho'(q_1) = 0$. Enfin, δ' est définie à partir de δ en posant: $\delta'(q,a) = \delta(q,a)$ pour tout $q \in Q \setminus \{q_{in}\}$ et toute lettre $a \in \{\bot,1\}$, $\delta'(q_{in},\bot) = \delta(q_{in},\bot) \wedge (q_1,1)$ et $\delta'(q_{in},1) = \delta(q_{in},1)$, $\delta(q_1,1) = (q_1,1)$ et $\delta(q_1,\bot) = ff$. Du fait de l'état q_1 le seul mot pouvant être accepté est $\bot 1^\omega$, et il est facile de voir qu'un mot accepté par $\mathcal{A}'$ l'est aussi par $\mathcal{A}$. On a donc prouvé le résultat qui suit.

Proposition 7.8 *Le problème de décider si une configuration $(p_{in},\bot)$ est gagnante pour Eve dans $\mathbb{G}$ se réduit polynomialement à décider le vide pour un automate bidirectionnel alternant de parité.*

Concernant le problème du vide pour les automates bidirectionnel alternant de parité, on a le résultat suivant.

Proposition 7.9 *Décider le vide pour un automate bidirectionnel alternant de parité est un problème* Pspace*-complet.*

La preuve de la proposition 7.9 occupe le reste du chapitre.

On considère un automate bidirectionnel alternant de parité $\mathcal{A} = \langle Q,A,q_{in},\delta,\rho\rangle$ sur les couleurs $\{1,\dots d\}$. On souhaite décider si le langage reconnu par $\mathcal{A}$ est vide. Pour cela, on construit un automate alternant de parité $\mathcal{B}$ reconnaissant le même langage. De plus $\mathcal{B}$ est polynomial dans la taille de $\mathcal{A}$.

La construction que l'on donne est une adaptation de la preuve de Vardi pour le cas des automates d'arbres [79] et une extension d'un résultat de Kupferman, Piterlan et Vardi [49, 71].

Enfin, à l'aide d'une analyse fine de la structure de $\mathcal{B}$, on montre que l'on peut tester si le langage reconnu par $\mathcal{B}$ est vide en espace polynomial, ce qui conclut la preuve.

Pour construire $\mathcal{B}$, on raisonne comme dans [79], et l'on se calque sur la trame de [71]

Définition 7.4 *Une* stratégie *pour $\mathcal{A}$ est une fonction τ de $\mathbb{N}$ dans $2^{Q\times\{-1,0,1\}\times Q}$. Pour tout sous-ensemble $\zeta \subseteq Q\times\{-1,0,1\}\times Q$, on pose $state(\zeta) = \{q \in Q \mid (q,i,q') \in \zeta, q' \in Q, i = -1,0,1\}$. La stratégie τ* est sur un mot *$u = a_0a_1\cdots$ si $q_{in} \in state(\tau(0))$, et pour tout $i \geq 0$ et tout état $q \in state(\tau(i))$, $\{(q',c) \mid (q,c,q') \in \tau(i)\} \models \delta(q,a_i)$.*

Définition 7.5 *Un* chemin *dans une stratégie τ est une suite finie ou infinie $(0,q_{in}),(i_1,q_1),(i_2,q_2)\dots$ de couples dans $\mathbb{N}\times Q$ tels que, soit le chemin est infini, et pour tout $j \geq 0$, il existe $c_j \in \{-1,0,1\}$ tel que $(q_j,c_j,q_{j+1}) \in \tau(i_j)$ et $i_{j+1} = i_j + c_j$, soit le chemin est une suite fini $(0,q_{in}),(i_1,q_1)\dots(i_m,q_m)$ et pour tout $0 \leq j < m$, il existe $c_j \in \{-1,0,1\}$ tel que $(q_j,c_j,q_{j+1}) \in \tau(i_j)$ et $i_{j+1} = i_j + c_j$, et $\delta(q_m,a_m) = tt$. Un chemin $(0,q_{in}),(i_1,q_1),(i_2,q_2)\dots$ est* acceptant *s'il est fini, ou s'il est infini et satisfait la condition de parité, c'est-à-dire que $\liminf\{\rho(q_i) \mid i \geq 1\}$ est paire. Enfin, une stratégie τ est* acceptante *si tout les chemins infinis dans τ sont acceptants.*

On a alors la proposition suivante.

Proposition 7.10 *[79] Un automate bidirectionnel alternant de parité accepte un mot si et seulement s'il possède une stratégie acceptante sur ce mot.*

On introduit alors la notion d'annotation qui exprime les diverses boucles (et leur couleur minimale, où l'on ignore la couleur de l'état de départ) pouvant intervenir lors d'un chemin dans la stratégie τ. Une *annotation* pour $\mathcal{A}$ de la stratégie τ (sur un mot u) est une application η de $\mathbb{N}$ dans $2^{Q\times\{1,\dots,d\}\times Q}$ qui satisfait les conditions suivantes pour tout $i \in \mathbb{N}$:

1. si $(q,c,q') \in \eta(i)$ et $(q,c',q') \in \eta(i)$, alors $(q,\min(c,c'),q') \in \eta(i)$: si on a une boucle de couleur minimale c et une boucle de couleur minimale c' on a une boucle de couleur minimale $\min(c,c')$.
2. si $(q,0,q') \in \tau(i)$ alors $(q,\rho(q'),q') \in \eta(i)$: si on a une boucle triviale, la seule couleur est celle du dernier état.
3. si $i > 0$, $(q,-1,q') \in \tau(i)$, $(q',c,q'') \in \eta(i-1)$ et $(q'',1,q''') \in \tau(i-1)$, alors $(q,\min(\rho(q'),c,\rho(q''')),q''') \in \eta(i)$: on va à gauche, on boucle et l'on revient à droite.

4. si $(q,1,q') \in \tau(i)$, $(q',c,q'') \in \eta(i+1)$ et $(q'',-1,q''') \in \tau(i+1)$, alors $(q, \min(\rho(q'),c,\rho(q''')),q''') \in \eta(i)$: on va à droite, on boucle et l'on revient à gauche.
5. si $i > 0$, $(q,-1,q') \in \tau(i)$ et $(q',1,q'') \in \tau(i-1)$, alors $(q, \min(\rho(q'),\rho(q'')),q'') \in \eta(i)$: on va à gauche et l'on revient tout de suite.
6. si $(q,1,q') \in \tau(i)$ et $(q',-1,q'') \in \tau(i+1)$, alors $(q, \min(\rho(q'),\rho(q'')),q'') \in \eta(i)$: on va à droite et l'on revient tout de suite.

Un *chemin descendant* dans une annotation η d'une stratégie τ sur un mot $u = a_0a_1\cdots$ est une suite de triplets $(i_1,q_1,t_1),(i_2,q_2,t_2),\ldots$ où pour tout $j \geq 1$, $i_j \in \mathbb{N}$, $q_j \in Q$ et t_j est un élément de $\tau(i_j)$ ou de $\eta(i_j)$, satisfaisant :

– si t_j est un élément de $\tau(i_j)$, $t_j = (q_j,1,q_{j+1})$ et $i_{j+1} = i_j + 1$. Dans ce cas, on dit que la couleur de (i_j,q_j,t_j) est $\rho(q_{j+1})$.
– si t_j est un élément de $\eta(i_j)$, $t_j = (q_j,c,q_{j+1})$ et $i_{j+1} = i_j$. Dans ce cas, on dit que la couleur de (i_j,q_j,t_j) est c.

Un chemin descendant peut être fini, s'il se termine par un triplet (i_m,q_m,t_m) avec $t_m = (q,c,q)$ (on termine par une boucle) ou s'il se termine par un triplet de la forme (i_m,q_m,t_m) avec $\delta(q_m,a_{i_m}) = \not\Vdash$. Il est *acceptant* si l'on est dans le premier cas avec c pair, ou si l'on est dans le second cas. Enfin, un chemin descendant infini est acceptant s'il satisfait la condition de parité, c'est-à-dire si la plus petite couleur (au sens défini ci-dessus) infiniment répétée est paire.

Enfin, une annotation η est acceptante si tous ses chemins descendants sont acceptants. On a alors la caractérisation suivante.

Proposition 7.11 *[79]Un automate bidirectionnel alternant de parité accepte un mot si et seulement s'il existe une stratégie sur le mot et une annotation acceptante de la stratégie.*

Remarque 7.4 La notion de chemin descendant est en fait l'analogue de la notion de M/B factorisation introduite dans l'étude des conditions de parité pour les jeux sur des graphes de processus à pile (voir chapitre 5). La proposition précédente peut alors être interprétée comme l'analogue de la proposition 5.1 disant qu'une stratégie est gagnante si pour toute contre-stratégie, la M/B factorisation de la partie est telle que la plus petite couleur apparaissant infiniment souvent dans la suite des couleurs des facteurs est paire.

On peut alors définir un automate alternant $\mathcal{B}$ de parité équivalent à $\mathcal{A}$. L'automate $\mathcal{B}$ va lire un mot de A ainsi qu'une stratégie de $\mathcal{A}$ sur ce mot, et une annotation de cette stratégie. Il accepte si et seulement si l'annotation est acceptante.

On introduit donc deux nouveaux alphabets : $\Delta_Q^s = 2^{Q\times\{-1,0,1\}\times Q}$ (pour les stratégies) et $\Delta_Q^a = 2^{Q\times\{1,\ldots,d\}\times Q}$ (pour les annotations). Enfin, on considère l'alphabet $A' = A \times \Delta_Q^s \times \Delta_Q^a$, et les trois projections naturelles p_1, p_2 et p_3 de A' sur A, Δ_Q^s et Δ_Q^a, projections qui définissent des morphismes lettre à lettre de A'^* dans A^*, ${\Delta_Q^s}^*$ et ${\Delta_Q^a}^*$. Enfin, l'automate $\mathcal{B}$ est l'intersection de deux automates $\mathcal{B}_1$ et $\mathcal{B}_2$. Sur une entrée u, l'automate $\mathcal{B}_1$ vérifie que $p_3(u)$ est une annotation de la stratégie $p_2(u)$, tandis que $\mathcal{B}_2$ vérifie que tous les chemins descendants sont acceptants. Un point crucial pour la suite est que $\mathcal{B}_1$ n'a pas de condition d'acceptation (il peut juste bloquer) et que $\mathcal{B}_2$ est purement universel (tous ses états sont universels).

Description de $\mathcal{B}_1$: Afin de vérifier que $p_2(u)$ est une stratégie sur u et afin de vérifier les conditions 1 et 2 pour que $p_3(u)$ soit une annotation de $p_2(u)$, il suffit de vérifier des conditions locales, et l'on peut alors restreindre l'alphabet A' de telles sortes à ce que ces dernières soient toujours satisfaites. On se concentre donc sur les conditions 3 à 6 pour $p_3(u)$.

On pose $\mathcal{B}_1 = \langle Q_1, A', \{q^1_{in}\}, \delta_1, \rho_1\rangle$ où Q_1 est composé des ensembles d'états suivants :

- deux états $\{q^1_{in}, q^1_1\}$ qui permettent d'initier et de propager la vérification des différentes conditions. L'état q^1_{in} vérifie que q_{in} est dans $state(\zeta)$ si la lettre lue est (a,ζ,μ). L'état q^1_1 sert à vérifier récursivement les autres conditions dans le reste du mot.
- un ensemble d'états $\{C\} \times Q \times \{\in, \notin\}$, qui servent à vérifier la consécution de la stratégie, à savoir que s'il y a un état $(q,1,q')$ dans la stratégie courante, il doit y avoir un état $(q',\_,\_)$ dans la stratégie suivante, et s'il n'y a pas d'état de la forme $(q,\_,\_)$ dans la stratégie courante, il ne doit pas y avoir $(q',-1,q)$ dans la stratégie suivante.
- un ensemble d'états $\{A\} \times Q \times \{1,\ldots,d\} \times Q \times \{\in, \notin\}$, qui représentent des triplets (q,c,q') de l'annotation qui doivent appartenir $(\in)$ ou non $(\notin)$ à la troisième composante de la lettre courante.
- un ensemble d'états $\{S\} \times Q \times \{-1\} \times Q \times \{\notin\}$, qui représentent des triplets $(q,-1,q')$ de la stratégie qui ne doivent pas se trouver dans la seconde composante de la lettre courante.

La fonction de coloriage ne sert à rien ici, et l'on considère que c'est la fonction constante égale à 2. Dès lors, toute branche infinie dans le calcul vérifie la condition de parité.

La fonction de transition δ_1 est donnée par :

- $\delta_1((C,q,\in),(a,\zeta,\mu))$ vaut tt si $q \in state(\zeta)$ ou si $\delta(q,a) = tt$, et vaut ff sinon.
- $\delta_1((C,q,\notin),(a,\zeta,\mu))$ vaut ff s'il existe q' tel que $(q',-1,q) \in \zeta$, et vaut tt sinon.
- $\delta_1((A,q_1,c,q_2,\in),(a,\zeta,\mu))$ vaut tt si $(q_1,c,q_2) \in \mu$, et vaut ff sinon.
- $\delta_1((A,q_1,c,q_2,\notin),(a,\zeta,\mu))$ vaut tt si $(q_1,c,q_2) \notin \mu$, et vaut ff sinon.
- $\delta_1((S,q_1,i,q_2,\notin),(a,\zeta,\mu))$ vaut tt si $(q_1,i,q_2) \notin \zeta$, et vaut ff sinon.
- on pose $consec(a,\zeta) = \{q \in Q \mid q \notin state(\zeta) \text{ et } \delta(q,a) \neq tt\}$, pour tout $a \in A$ et tout $\zeta \in Q \times \{-1,0,1\} \times Q$. La consécution de la stratégie est exprimée par la formule $\Upsilon^{cons}(a,\zeta,\mu) = \bigwedge_{q \in consec(a,\zeta)} (c,q,\notin) \wedge \bigwedge_{(q',A,q)\in\zeta} (c,q,in)$.
- pour la condition 3, si l'on pose $\varphi_1 = (q,-1,q') \in \tau(i)$, $\varphi_2 = (q',c,q'') \in \eta(i-1)$, $\varphi_3 = (q'',1,q''') \in \tau(i-1)$ et $\varphi_4 = (q, \min(c,\rho(q'),\rho(q''')),q''') \in \eta(i)$, la condition s'exprime par la formule logique $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \Rightarrow \varphi_4$, qui est équivalente à $\varphi_2 \wedge \varphi_3 \Rightarrow \varphi_4 \vee \neg\varphi_1$. D'où l'expression
  $\Upsilon^3(a,\zeta,\mu) = \bigwedge_{(q',c,q'')\in\mu} \bigwedge_{(q'',1,q''')\in\zeta} \bigwedge_{q\in Q} [(A,q,c',q''',\in) \vee (S,q,-1,q',\notin)]$, où $c' = \min(c,\rho(q'),\rho(q'''))$.
- de même, pour la condition 4, si l'on pose $\varphi_1 = (q,1,q') \in \tau(i)$, $\varphi_2 = (q',c,q'') \in \eta(i+1)$, $\varphi_3 = (q'',-1,q''') \in \tau(i+1)$ et $\varphi_4 = (q, \min(c,\rho(q'),\rho(q''')),q''') \in \eta(i)$,

la condition s'exprime par la formule logique $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \Rightarrow \varphi_4$, qui est équivalente à $\varphi_1 \wedge \neg\varphi_4 \Rightarrow \neg\varphi_2 \vee \neg\varphi_3$. D'où l'expression :
$\Upsilon^4(a,\zeta,\mu) = \bigwedge_{(q,1,q')\in\zeta} \bigwedge_{(q,c',q''')\in\mu} \bigwedge_{q''\in Q} [(A,q',c,q'', \notin) \vee (S,q'', -1,q''', \notin)]$, où $c' = \min(c,\rho(q'),\rho(q'''))$.

– de même pour la condition 5, on pose :
$\Upsilon^5(a,\zeta,\mu) = \bigwedge_{(q,1,q'')\in\zeta} \bigwedge_{q\in Q} [(A,q,c,q'', \in)\vee(S,q,-1,q', \notin)]$, où $c = \min(\rho(q'),\rho(q''))$.
– de même pour la condition 6, on pose :
$\Upsilon^5(a,\zeta,\mu) = \bigwedge_{(q,1,q')\in\zeta} \bigwedge_{(q,c,q'')\notin\mu} (S,q,-1,q'', \notin)$, où $c = \min(\rho(q'),\rho(q''))$.
– pour l'initialisation, $\delta_1(q^1_{in},(a,\zeta,\mu))$ vaut $q^1_1 \wedge \Upsilon^c(a,\zeta) \wedge \Upsilon^3(a,\zeta,\mu) \wedge \Upsilon^4(a,\zeta,\mu) \wedge \Upsilon^5(a,\zeta,\mu) \wedge \Upsilon^6(a,\zeta,\mu)$ si $q_0 \in \mathit{state}(\zeta)$, et *ff* sinon.
– pour la propagation, on pose enfin, $\delta_1(q^1_1,(a,\zeta,\mu)) = q^1_1 \wedge \Upsilon^c(a,\zeta) \wedge \Upsilon^3(a,\zeta,\mu) \wedge \Upsilon^4(a,\zeta,\mu) \wedge \Upsilon^5(a,\zeta,\mu) \wedge \Upsilon^6(a,\zeta,\mu)$.

On vérifie alors que $\mathcal{B}_1$ réalise bien ce que l'on souhaitait.

Description de $\mathcal{B}_2$: On rappelle que $\mathcal{B}_2$ sert à vérifier que tous les chemins descendants sont acceptants. Dès lors, il va être équipé d'une condition de parité et sera purement universel. Plus précisément, $\mathcal{B}_2 = \langle Q\times\{1,\ldots,d\},A',(q_{in},\rho(q_{in})),\delta_2,\rho_2\rangle$, où l'on pose :

– pour tout état (q,c) et toute lettre (a,ζ,μ), $\delta_2((q,c),(a,\zeta,\mu))$ vaut *ff* s'il existe une couleur impaire c' telle que $(q,c',q) \in \mu$ ou s'il existe une couleur impaire c', une couleur quelconque c'' et un état $q' \in Q$, tels que $(q,c'',q') \in \mu$ et $(q',c',q') \in \mu$.
Sinon, $\delta_2((q,c),(a,\zeta,\mu)) = \bigwedge_{(q,1,q')\in\zeta} (q',\rho(q')) \wedge \bigwedge_{(q,c',q')\in\mu} \bigwedge_{(q',1,q'')\in\zeta} (q'', \min(c',\rho(q'')))$
– pour tout état (q,c), $\rho_2((q,c)) = c$.

Il est alors facile de voir que $\mathcal{B}_2$ accepte si et seulement si tous les chemins descendants sont acceptants.

On a donc $L(\mathcal{B}) = L(\mathcal{B}_1) \cap L(\mathcal{B}_1)$. Si l'on note $n = |Q|$, $n_1 = |Q_1|$ et $n_2 = |Q_2|$, on a $n_1 = \mathcal{O}(nd)$ et $n_2 = \mathcal{O}(nd)$. Par ailleurs, l'automate $\mathcal{B}_1$ n'a pas de condition d'acceptation (tout calcul infini est acceptant). Quant à $\mathcal{B}_2$, il est purement universel.

L'automate $\mathcal{B}_2$, étant purement universel de parité, l'automate $\overline{\mathcal{B}_2}$ obtenu par dualisation est un automate non déterministe de parité possédant n_2 états et faisant intervenir d couleurs. Il est facile à présent de construire un automate de Büchi non déterministe $\overline{\mathcal{B}'_2}$ reconnaissant le même langage que $\overline{\mathcal{B}_2}$ et possédant $\mathcal{O}(n_2 d)$ états (voir par exemple [55]). Enfin, en dualisant $\overline{\mathcal{B}_2}$, on obtient un automate de co-Büchi $\mathcal{B}'_2$ purement universel reconnaissant le même langage que $\mathcal{B}_2$ et possédant $\mathcal{O}(n_2 d)$ états.

En prenant l'intersection de $\mathcal{B}'_2$ et de $\mathcal{B}_1$, on obtient un automate alternant muni d'une condition de co-Büchi $\mathcal{B}$ possédant $\mathcal{O}(n_2 d + n_1)$ états et reconnaissant le langage $L(\mathcal{B}_1) \cap L(\mathcal{B}_2) = L(\mathcal{A})$.

Enfin, décider le vide pour un automate alternant muni d'une condition de co-Büchi étant un problème PSPACE (voir par exemple [51]), on en conclut facilement que l'on peut décider si $L(\mathcal{A})$ est vide en PSPACE.

On a donc prouvé le résultat suivant.

Théorème 7.3 *Décider si une position de pile vide est gagnante dans un jeu de parité sur un graphe d'automate à compteur est dans* PSPACE.

7.2.3 Borne inférieure

Le théorème 7.3 donne une borne supérieure pour la décision du gagnant dans un jeu sur un graphe de processus à compteur. Il est alors naturel de chercher une borne inférieure. Nous donnons ici une première borne inférieure en montrant que, même dans le cas d'une condition d'accessibilité, le problème est NP-dur, en réduisant le problème SAT. Par symétrie du problème, il vient immédiatement que le problème est également coNP-dur. En adaptant la réduction utilisée pour réduire SAT, on montre que l'on peut réduire le problème SAT-UNSAT qui est complet pour la classe DP (définie ci-dessous). Ce dernier résultat, plus fort que les deux autres, est plus satisfaisant. En effet, le fait que DP soit fermée par complémentaire est plus en accord avec le caractère symétrique du problème de décision du gagnant dans un jeu.

On commence donc par la définition de la classe DP et du problème SAT-UNSAT (pour plus de détails, voir [69]).

Définition 7.6 *Un langage L est dans la classe de complexité* DP *si et seulement s'il existe un langage L_1 dans* NP *et un langage L_2 dans* coNP *tels que $L = L_1 \cap L_2$.*

Définition 7.7 *Le problème* SAT-UNSAT *est le suivant : étant données deux formules booléennes ψ_1 et ψ_2, sous forme normale conjonctive avec trois littéraux par clause, a-t-on que ψ_1 est satisfiable et ψ_2 est non satisfiable? Le problème* SAT-UNSAT *est complet pour la classe* DP.

On peut alors énoncer notre résultat.

Théorème 7.4 *Décider si une position de pile vide est gagnante dans un jeu d'accessibilité sur un graphe de processus à compteur est dur pour les classes de complexités suivantes :* NP, coNP *et* DP.

Preuve. On commence par montrer que le problème est NP-dur. Pour cela on donne une réduction polynomiale du problème SAT dans sa variante où la formule donnée est sous forme normale conjonctive avec trois littéraux par clause. On considère donc une telle formule ψ, sur un ensemble de variables $X = \{x_1, \ldots, x_k\}$. On note $\psi = C_1 \wedge C_2 \wedge \cdots \wedge C_h$, et l'on pose $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ pour tout $i = 1, \ldots, h$ avec $l_{i,k} \in \{x, \overline{x} \mid x \in X\}$, pour $k = 1,2,3$. Enfin, on désigne, par ρ_i le i-ème nombre premier, pour tout $i \geq 1$.

Une valuation de X étant une application de X dans $\{0,1\}$, on la représente comme un vecteur de $\{0,1\}^k$. On donne une application τ de $\mathbb{N}$ dans $\{0,1\}^k$ associant à toute entier une valuation de X. Pour tout $n \geq 0$ on pose $\tau(n) = (b_1, b_2, \ldots, b_k)$ où b_j vaut 0 si $n = 0 \mod \rho_j$ et vaut 1 sinon. Par exemple, si $k = 5$, $\tau(132) = (0,0,1,1,0)$. Le lemme chinois montre que τ est une surjection, ce qui permet de coder toute valuation de X par un entier.

On considère le jeu suivant : Eve donne un entier n qui code une valuation de ψ, puis Adam choisit une clause C_i qu'il pense ne pas être satisfaite par la valuation précédente. Eve donne alors un littéral de C_i et Adam vérifie enfin si la valuation précédente évalue ce dernier à 1. Si c'est le cas, Eve remporte la partie, sinon c'est Adam qui gagne. Il est clair qu'Eve possède une stratégie gagnante dans ce jeu, si et seulement si ψ est satisfiable.

Il reste donc à expliquer comment coder un tel jeu par un jeu d'accessibilité sur un graphe de processus à compteur. On considère donc un processus à compteur et un jeu d'accessibilité dessus, que l'on ne décrit pas en détail. On s'intéresse aux parties commençant depuis la configuration $(q_{in},\bot)$. C'est Eve qui joue depuis cet état. Elle incrémente alors le compteur et peut soit rester dans q_{in}, soit passer dans l'état q_{cc}. Dans ce dernier cas, on est alors dans une configuration $(q_{cc},1^n\bot)$ codant la valuation $\tau(n)$, et c'est Adam qui joue. Il peut alors appliquer une règle $skip(q_{C_i})$ pour tout $i = 1,\ldots,h$, signifiant ainsi qu'il souhaite vérifier que C_i est satisfaite par $\tau(n)$. Depuis la configuration $(q_{C_i},1^n\bot)$, Eve applique une règle $skip(q_{l_{i,j}})$ pour $j \in \{1,2,3\}$, signifiant que le littéral $l_{i,j}$ est satisfait par $\tau(n)$. On effectue ensuite une transition $skip(m_0^{\rho_k})$ si $l_{i,j} = x_k$ et $skip(\overline{m}_0^{\rho_k})$ si $l_{i,j} = \overline{x_k}$. Depuis la configuration $(m_0^{\rho_k},1^n\bot)$ on dépile, tant que la pile est non vide, tout en comptant modulo ρ_k : on va donc successivement dans les états $m_1^{\rho_k},m_2^{\rho_k},\ldots,m_{\rho_k-1}^{\rho_k},m_0^{\rho_k},m_1^{\rho_k}\ldots$ Enfin, on atteint une configuration $(m_l^{\rho_k},\bot)$ depuis laquelle on applique la règle $skip(q_w)$ si $l = 0$ et $skip(q_l)$ sinon. Les états q_w et q_l sont quant à eux bloquants. Dans le cas où $l_{i,j} = \overline{x_k}$, la partie se passe de la même façon, sauf qu'on va à la fin dans $(q_w,\bot)$ depuis $(\overline{m}_l^{\rho_k},\bot)$ si $l \neq 0$ et dans $(q_l,\bot)$ sinon.

Enfin, il y a un unique état final, à savoir q_w. Il est alors clair qu'Eve possède une stratégie gagnante depuis $(p_{in},\bot)$ si et seulement si ψ est satisfiable.

Il reste à établir le caractère polynomial de la réduction. Pour cela, deux points sont à préciser : le fait que le processus à compteur soit de taille polynomiale, et le fait que l'on puisse le construire en temps polynomial. Le nombre d'états du processus à compteur est $\mathcal{O}(\sum_{i=1}^{k}\rho_i)$, qui est polynomial en k, car ρ_i est borné par un polynôme en i (du fait que le nombre $\pi(x)$ de nombres premiers inférieurs à x évolue en $\frac{x}{\ln(x)}$). Pour ce qui est de la capacité à construire le processus, il suffit de remarquer que l'on peut déterminer les k premiers nombres premiers en temps polynomial en k, ce qui ne pose aucun problème (un simple crible suffit).

Pour ce qui est du caractère coNP-dur, la preuve est obtenue en réduisant le problème UNSAT. On inverse alors le rôle des deux joueurs : c'est Adam qui donne la valuation et c'est Eve qui choisit la clause à vérifier. Enfin, Adam choisit le littéral et si celui-ci n'est pas satisfait, Eve gagne sinon c'est Adam.

Pour ce qui est du caractère DP-dur, on réduit SAT-UNSAT. On a donc deux formules ψ_1 et ψ_2. Au début, Adam choisit si l'on vérifie que ψ_1 est satisfiable ou si l'on vérifie que ψ_2 n'est pas satisfiable. Dans le premier cas, on joue comme pour la première réduction, et dans le second cas, on joue comme pour la seconde réduction. □

7.2.4 Conséquences

Complexité du model-checking du μ-calcul pour un processus à compteur

Nous donnons maintenant une borne inférieure et une borne supérieure pour le problème du model-checking du μ-calcul pour un processus à compteur. Pour une définition du μ-calcul, on pourra consulter [6, 83].

Dans [83, 84], I. Walukiewicz prouve le résultat suivant :

Théorème 7.5 *Le problème du model-checking du μ-calcul sur un processus à pile est polynomialement équivalent à la décision du gagnant dans un jeu de parité sur un graphe de processus à pile.*

La réduction du problème de model-checking vers le jeu modifie seulement l'ensemble des états, et laisse intact l'alphabet de pile. Quant à la réduction réciproque, elle ne considère pas la pile. Dès lors, on a le corollaire suivant du théorème 7.5

Corollaire 7.1 *Le problème du model-checking du μ-calcul sur un processus à compteur est polynomialement équivalent à la décision du gagnant dans un jeu de parité sur un graphe de processus à compteur.*

On déduit des bornes données dans les sections 7.2.2 et 7.2.3, le corollaire suivant.

Corollaire 7.2 *Le problème du model-checking du μ-calcul sur un processus à compteur est dans* Pspace. *De plus, c'est un problème* DP*-dur.*

Combinaisons booléennes d'une condition de parité et d'une condition sur la hauteur de pile

Dans le paragraphe 8.2 du chapitre 8, on considère un jeu sur un graphe de processus à pile muni d'une condition de gain qui est une combinaison booléenne d'une condition de parité et d'une condition sur la hauteur de pile. On montre comment réduire le problème de décision du gagnant dans un tel jeu au problème de décision du gagnant dans un jeu de parité sur un graphe de processus à compteur exponentiellement plus grand. Le théorème 7.3 permet alors de déduire un algorithme en Expspace pour le premier problème. Ces constructions sont exposées en détail dans le paragraphe 8.2.

Chapitre 8

D'autres conditions de gain

Sommaire

Dans ce chapitre, on considère des conditions naturelles pour la vérifications de systèmes. Les premières sont celles données par un langage ω-VPL et concernent les jeux sur des graphes de VPP. Les résultats présentés sont ceux obtenus avec C. Löding et P. Madhusudan dans [56].

On revient ensuite aux jeux sur un graphe de processus à pile. Si ce dernier modélise un système ouvert, il peut être intéressant de rechercher un contrôleur permettant de satisfaire une propriété régulière mais aussi une contrainte sur la hauteur de pile, comme un bornage. En terme de jeux, cela revient à chercher une stratégie gagnante pour une condition de gain qui est une combinaison booléenne d'une condition de parité et d'une condition sur la hauteur de pile. C'est l'objet du second paragraphe. On y donne une solution qui utilise les résultats obtenus dans le chapitre précédent pour les jeux de parité sur un graphe de processus à compteur. On présente également les constructions obtenues avec A.J. Bouquet et I. Walukiewicz dans [13].

8.1 Jeux sur des graphes de VPP muni d'une condition ω-VPL

8.1.1 Définitions et motivations

Dans [4], les auteurs considèrent l'alphabet d'actions $\widetilde{A} = \langle\{c\},\{r\},\emptyset\rangle$. Un mot sur l'alphabet A est *infiniment borné* si, lorsqu'il est lu par un VPA, la pile de se dernier visite infiniment souvent une hauteur de pile. Le langage des mots infiniment bornés est noté L_{repbdd}. Ce langage est ω-VPL, et il en va de même pour son complémentaire, puisque les langages ω-VPL sont fermés par complémentation [4].

Considérons maintenant un graphe de jeu $\mathcal{G}$ sur un VPP, d'alphabet d'actions A. Considérons le jeu $\mathbb{G}$ sur $\mathcal{G}$ muni de la condition externe $\overline{L_{repbdd}}$: $\mathbb{G}$ n'est autre que le jeu d'explosion stricte sur le graphe de jeu $\mathcal{G}$.

Les langages ω-VPL étant fermés par opérations booléennes, on peut donc spécifier des comportements parlant à la fois de la hauteur de pile et de conditions régulières. En fait, R. Alur, K. Etessami et P. Madhusudan ont introduit dans [1] une extension de la logique LTL, appelée CARET, permettant de parler d'un appel et de son retour associé. Par exemple, on peut exprimer des propriétés du type *si une propriété p est vraie lors d'un appel à un module, le module doit terminer et une fois celui-ci fini, une propriété q doit être vraie*. Un langage définissable par une formule CARET est en réalité un langage ω-VPL [4].

Dans [4], les auteurs montrent comment vérifier qu'un système décrit par un VPP vérifie une propriété ω-VPL. En d'autres termes, ils montrent comment décider le gagnant dans un jeu à un joueur sur un graphe de VPP muni d'une condition externe décrite par un langage ω-VPL.

Dans [56], avec C. Loeding et P. Madhusudan, nous avons considéré le cas des jeux à deux joueurs. On présente dans ce qui suit les résultats que nous avons obtenus. La principale difficulté provient du fait que le modèle d'automates définissant les langages ω-VPL est non déterministe. Dans un premier temps, on propose un modèle déterministe de VPA que l'on équipe d'une nouvelle condition d'acceptation, la condition de *parité en escalier*. Ensuite, on montre comment réduire notre problème à décider le gagnant dans un jeu de parité en escalier sur un graphe de processus à pile, et l'on conclut en utilisant les résultats de la partie 5. On donne une borne supérieure et inférieure pour ce problème. On termine enfin par des considérations sur les stratégies gagnantes et sur la complexité borélienne des langages ω-VPL.

8.1.2 Déterminisation des ω-VPA

Dans [4], les auteurs montrent qu'un VPA non déterministe de Büchi ne peut pas toujours être transformé en un VPA déterministe de Muller reconnaissant le même langage. Un exemple de tel langage est l'ensemble des mots qui lorsqu'ils sont lus par un VPA sont tels que la pile de ce dernier répète infiniment souvent un niveau. Un automate non déterministe va pouvoir deviner quel sera le niveau infiniment souvent visité (et passer par un état final à chaque visite) tandis qu'un automate déterministe, même muni d'une condition de Muller ne pourra repérer ce phénomène.

Nous introduisons ici une nouvelle condition d'acceptation pour les VPA, et nous montrons que les VPA déterministes équipés de cette condition d'acceptation ont le même pouvoir d'expression que les VPA non déterministes de Büchi.

L'idée est de ne pas évaluer la condition d'acceptation sur la suite complète des états visités lors d'un calcul, mais seulement sur une sous-suite, correspondant à des positions minimales pour la hauteur de pile. Ainsi, la sous-suite de configurations obtenues sera croissante au sens large pour la hauteur de pile.

Nous commençons par quelques notations. Pour tout mot infini $u = a_0a_1a_2\cdots$, sur un alphabet quelconque A, et pour tout $X \subseteq \mathbb{N}$, on définit le mot $u|_X$, fini ou infini, sur l'alphabet A, par $u|_X = a_{n_1}a_{n_2}a_{n_3}\cdots$, où $X = \{n_1 < n_2 < n_3 < \cdots\}$.

Pour tout mot $u \in A^*$ sur un alphabet d'actions A, on définit la *hauteur de pile induite par* u, $sh(u)$ par : $sh(\varepsilon) = 0$ et,

$$\text{pour tout } a \in A,\ \begin{cases} sh(ua) = sh(u) & \text{si } a \in A_{int} \\ sh(ua) = sh(u) + 1 & \text{si } a \in A_c \\ sh(ua) = \max\{sh(u) - 1,0\} & \text{si } a \in A_r. \end{cases}$$

Enfin, pour tout mot infini $u \in A^\omega$, on note $Steps_u = \{n \in \mathbb{N} \mid \forall m \geq n : sh(u\restriction_m) \geq sh(u\restriction_n)\}$. Cet ensemble est toujours infini.

On appelle L_{wmw} l'ensemble des mot minimaux bien parenthésés, c'est-à-dire les mots de la forme $cur \in A^*$ où $r \in A_r$ est le retour correspondant à $c \in A_c$. Formellement, $c \in A_c$, $r \in A_r$, $sh(u) = 0$ et $sh(u') > 0$ pour tout préfixe strict u' de u. Tout mot infini $u \in A^\omega$ se factorise de façon unique en $u = u_1u_2u_3\cdots$, où chaque facteur $u_i \in L_{wmw} \cup A_c \cup A_r$. On établit facilement que si $u_i = c$ pour un appel $c \in A_c$, alors il n'existe pas d'indice $j \geq i$ tel que $u_j \in A_r$. De plus, pour tout $n \in \mathbb{N}$, $n \in Steps_u$ si et seulement s'il existe un $i \geq 0$ tel que $|u_1 \cdots u_i| = n$.

Etant donné un calcul d'un VPA sur un mot u, on considère la suite ξ des états du VPA au cours du calcul. La condition de parité en escalier va considérer la sous-suite $\xi|_{Steps_u}$ pour décider si le mot est accepté ou non. Si $\xi|_{Steps_u}$ vérifie une condition de parité, u est accepté sinon il est rejeté.

Définition 8.1 (STVPA) *Un automate non déterministe à pile avec visibilité muni d'une condition de parité en escalier, ou* STVPA*, est un septuplet* $\mathcal{A} = \langle Q,A,\Gamma,\bot,I,\rho,\delta\rangle$ *possédant les mêmes composantes qu'un* VPA *muni d'une condition de parité. En particulier,* ρ *est une application de* Q *dans un ensemble fini de couleurs* $C \subset \mathbb{N}$*. Un calcul d'un* STVPA *sur un mot* $u \in A^\omega$ *est acceptant si et seulement si la suite* $\xi = q_0q_1\cdots \in Q^\omega$ *des états au cours du calcul est telle que* $\liminf\{\rho(q_{n_i}) \mid n_i \in Steps_u\}$ *est paire. Le langage accepté par* $\mathcal{A}$ *est noté* $L(\mathcal{A})$*.*

Considérons maintenant un VPA non déterministe muni d'une condition de Büchi $\mathcal{A} = \langle Q,A,\Gamma,\bot,I,F,\delta\rangle$. Nous expliquons comment construire une STVPA déterministe qui reconnaît le même langage que $\mathcal{A}$. Pour cela, considérons un mot $u \in A^\omega$, ainsi que sa factorisation $u = u_1u_2u_3\cdots$ sur $L_{wmw} \cup A_c \cup A_r$. Dans un STVPA qui va lire u, la condition de parité ne sera évaluée qu'aux états atteints après avoir lu un facteur u_i. Dès lors, pour simuler $\mathcal{A}$, on va devoir *résumer* le comportement de ce dernier sur les facteurs u_i. Pour cela, on code la transformation qu'induit un de ces facteurs sur l'ensemble des états de $\mathcal{A}$.

Plus précisément, on considère l'ensemble des transformations $\mathcal{T}_Q = 2^{Q\times\{0,1\}\times Q}$. L'ensemble $\mathcal{T}_Q$ est naturellement muni d'une loi de composition que l'on note $\circ$.

La transformation T_{u_i}, induite par un facteur u_i, est donnée par : $(q,f,q') \in T_{u_i}$ si et seulement s'il existe un calcul de $\mathcal{A}$, d'étiquette u_i, depuis $(q,\perp)$ jusqu'à une configuration (q',σ) pour un $\sigma \in \Gamma^*$, et qui visite un état final si et seulement si $f = 1$. Le fait que la définition précédente soit fondée sur un calcul depuis une configuration de pile vide ne pose pas de problème. En effet, si u_i est un appel $c \in A_c$, la transition ne dépend pas du sommet de pile, si u_i est un mot bien parenthésé minimal, on ne dépile jamais depuis la pile vide, et enfin, si u_i est un retour $r \in A_r$, on sait que lors du calcul de $\mathcal{A}$ sur u on est dans une configuration de pile vide lorsque l'on lit r.

Considérons une suite de transformations $\tau = T_1 T_2 \cdots \in \mathcal{T}_Q^\omega$. La suite τ est dite *acceptante* si et seulement s'il existe une suite d'états $q_0 q_1 \cdots \in Q^\omega$ telle que :

- $q_0 \in I$;
- pour tout $i \geq 0$, $(q_i,f_i,q_{i+1}) \in T_{i+1}$ avec $f_i \in \{0,1\}$;
- il existe une infinité d'indices i tel que $f_i = 1$.

Maintenant, pour tout mot $u \in A^\omega$ de factorisation $u_1 u_2 \cdots$, on note $\tau_u \in \mathcal{T}_Q^\omega$, la suite $T_{u_1} T_{u_2} \cdots$. Il est alors facile de voir que u est accepté par $\mathcal{A}$ si et seulement si la suite τ_u est acceptante.

Par ailleurs, l'ensemble des suites acceptantes est un langage ω-régulier, et peut donc être reconnu par un automate déterministe de parité $\mathcal{A}_\mathcal{T} = \langle S,\mathcal{T}_Q,s_{in},\rho_\mathcal{T},\delta_\mathcal{T}\rangle$. De plus, on peut choisir $\mathcal{A}_\mathcal{T}$ de sorte que le nombre d'états soit $2^{\mathcal{O}(|Q|\cdot\log|Q|)}$.

Afin de simuler l'automate $\mathcal{A}$, on construit un VPA déterministe $\mathcal{C}$ avec sortie tel qu'après avoir lu un mot fini u de factorisation $u_1 u_2 \ldots u_k$, $\mathcal{C}$ sort, lors de sa dernière transition, la transformation $T_{out} = T_{u_k}$.

Afin de construire l'automate $\mathcal{C}$, on définit :

- pour tout appel $c \in A_c$ et toute lettre $\gamma \in (\Gamma \setminus \perp)$, on pose $T_{c,\gamma} = \{(q,f,q') \mid push(q',\gamma) \in \delta(q,\_,c)$ et $f = 1$ ssi $\{q,q'\} \cap F \neq \emptyset\}$;
- pour tout retour A_r et toute lettre $\gamma \in (\Gamma \setminus \perp)$, on pose $T_{r,\gamma} = \{(q,f,q') \mid pop(q') \in \delta(q,\gamma,c)$ et $f = 1$ ssi $\{q,q'\} \cap F \neq \emptyset\}$;
- pour tout retour A_r, on pose $T_{r,\perp} = \{(q,f,q') \mid skip(q') \in \delta(q,\perp,c)$ et $f = 1$ ssi $\{q,q'\} \cap F \neq \emptyset\}$.

On peut alors définir $\mathcal{C}$:

- l'ensemble des états de $\mathcal{C}$ est $\mathcal{T}_Q$.
- l'alphabet de pile de $\mathcal{C}$ est $(\mathcal{T}_Q \times A_c) \cup \{\perp\}$.
- l'état initial est $Id_Q = \{(q,0,q) \mid q \in Q\}$.
- la fonction de transition $\delta_\mathcal{C}$ est illustrée par la figure 8.1.2 et elle est donnée par :
  - pour tout $a \in A_{int}$, $T_{out} = T_a$ et $\delta_\mathcal{C}(T,\_,a) = \{skip(T \circ T_{out})\}$.
  - pour tout $c \in A_c$, $T_{out} = T_c$ et $\delta_\mathcal{C}(T',\_,c) = \{push(Id_Q,(T',c))\}$.
  - pour tout $r \in A_r$, si (T',c) désigne le sommet de pile, $T_{out} = \bigcup_{\gamma\in\Gamma\setminus\{\perp\}} T_{c,\gamma} \circ T \circ T_{r,\gamma}$ et $\delta_\mathcal{C}(T,(T',c),r) = \{pop(T' \circ T_{out})\}$.
  - pour tout $r \in A_r$, si $\perp$ désigne le sommet de pile, $T_{out} = T_r$ et $\delta_\mathcal{C}(T,\perp,r) = \{skip(T \circ T_{out})\}$.
- après la lecture de chaque lettre, l'automate sort la transformation T_{out}.

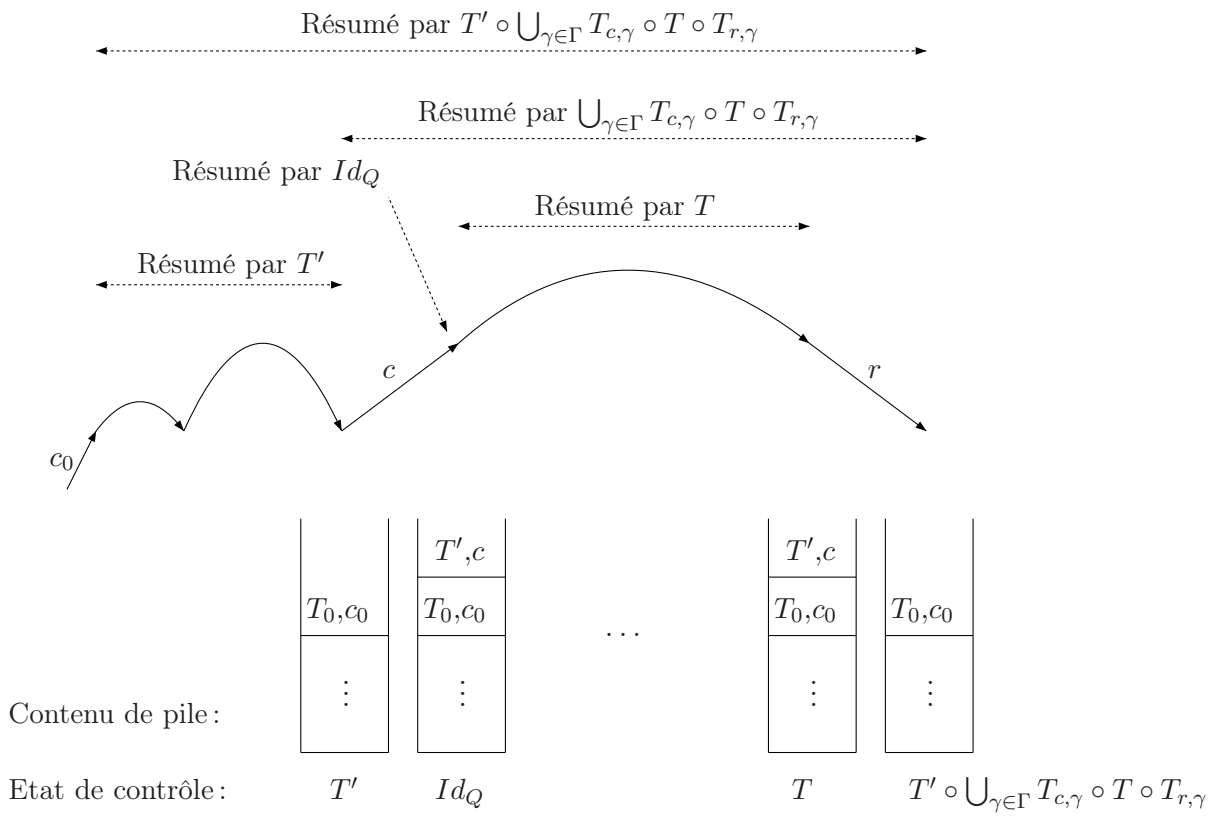


FIG. 8.1 – *Comportement de* $\mathcal{C}$

On vérifie sans problème qu'après avoir lu un mot u de factorisation $u_1 \cdots u_k$, la dernière transformation sortie par $\mathcal{C}$ est $\mathcal{T}_{u_k}$.

Maintenant, il n'est pas difficile d'obtenir à partir de $\mathcal{A}_\mathcal{T}$ et de $\mathcal{C}$, un StVPA déterministe $\mathcal{D}$ qui reconnaît le même langage que $\mathcal{A}$. L'ensemble des états de $\mathcal{D}$ est $\mathcal{T}_Q \times S$, et $\mathcal{D}$ va être construit de telle sorte à être, après avoir lu un mot fini u de factorisation $u = u_1 \cdots u_k$, dans un état dont la seconde composante est l'état atteint dans $\mathcal{A}_\mathcal{T}$ depuis s_{in} après avoir lu le mot $T_{u_1} \cdots T_{u_k}$. La fonction de coloriage sur $\mathcal{D}$ est induite par la fonction de coloriage $\rho_\mathcal{T}$ de $\mathcal{A}_\mathcal{T}$. Comme $\mathcal{D}$ reconnaît les transformations acceptantes, on aura $L(\mathcal{D}) = L(\mathcal{A})$.

Le StVPA $\mathcal{D}$ simule $\mathcal{C}$ sur sa première composante, tandis que la seconde composante est mise à jour grâce aux sorties de $\mathcal{C}$. En plus de l'information que $\mathcal{C}$ stocke dans la pile, lorsqu'un appel $a \in A_c$ est lu, $\mathcal{D}$ empile également l'état dans lequel il se trouvait au moment de la lecture de a. Lorsqu'un retour $r \in A_r$ est lu et que la pile est non vide, la seconde composante doit devenir $\delta_\mathcal{T}(s,T)$, où s est l'état de $\mathcal{A}_\mathcal{T}$ au moment où il a lu l'appel correspondant au retour r, et où T est la transformation induite par le facteur bien parenthésé que l'on vient de lire. L'état s est alors disponible en sommet de pile, tandis que T correspond à la sortie de $\mathcal{C}$: $\mathcal{D}$ peut donc mettre à jour correctement sa seconde composante.

Voici une description formelle de $\mathcal{D}$:

- les états de $\mathcal{D}$ sont $\mathcal{T}_Q \times S$.
- l'alphabet de pile est $(\mathcal{T}_Q \times A_c \times S) \cup \{\bot\}$.
- l'état initial est (Id_Q,s_{in}).
- la fonction de transition $\delta_\mathcal{D}$ est définie comme suit, où T_{out} désigne la sortie de $\mathcal{C}$ correspondant à la transition effectuée sur la première composante :
  - pour tout $a \in A_{int}$, $\delta_\mathcal{D}((T,s),\_,a) = \{skip((T',\delta_\mathcal{T}(s,T_{out})))\}$, où $\delta_\mathcal{C}(T,\_,a) = \{skip(T')\}$.
  - pour tout $c \in A_c$, $\delta_\mathcal{D}((T,s),\_,c) = \{push((Id_Q,\delta_\mathcal{T}(s,T_{out}),(T,c,s)))\}$.
  - pour tout $r \in A_r$ lorsque la pile est non vide, $\delta_\mathcal{D}((T,s),(T',c,s'),r) = \{pop((T'',\delta_\mathcal{T}(s',T_{out})))\}$, où $\delta_\mathcal{C}(T,(T',c),r) = \{pop(T'')\}$.
  - pour tout $r \in A_r$ lorsque la pile est vide, $\delta_\mathcal{D}((T,s),\bot,r) = \{skip((T'',\delta_\mathcal{T}(s',T_{out})))\}$, où $\delta_\mathcal{C}(T,\bot,r) = \{skip(T'')\}$.
- la fonction de coloriage est obtenue en étendant $\rho_\mathcal{T}$: $\rho_\mathcal{D}(T,s) = \rho_\mathcal{T}(s)$.

On a donc prouvé

Théorème 8.1 *[56] Pour tout* VPA *non déterministe de Büchi $\mathcal{A}$ sur un alphabet A, il existe un* StVPA *déterministe de parité $\mathcal{D}$ tel que $L(\mathcal{A}) = L(\mathcal{D})$. De plus, $\mathcal{D}$ peut être construit de sorte à avoir $2^{\mathcal{O}(n^2)}$ états, où n désigne le nombre d'états de $\mathcal{A}$.*

Réciproquement, on peut prouver sans grande difficulté que la condition de parité en escalier n'augmente pas le pouvoir de reconnaissance des VPA.

Théorème 8.2 *[56] Pour tout* StVPA *non déterministe $\mathcal{A}$ sur un alphabet A, il existe un* VPA *non déterministe de Büchi $\mathcal{A}'$ tel que $L(\mathcal{A}) = L(\mathcal{A}')$.*

8.1.3 Jeux

Considérons un VPP $\mathcal{P} = \langle Q,A,\Gamma,\bot,\Delta\rangle$ et une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de Q. On appelle $\mathcal{G} = ((V,E),V_{\mathbf{E}},V_{\mathbf{A}})$ le graphe de jeu engendré par $\mathcal{P}$ et la partition précédente. On considère enfin un langage ω-VPL L sur l'alphabet d'actions A, et l'on appelle $\mathbb{G} = (\mathcal{G},L)$ le jeu sur $\mathcal{G}$ muni de la condition externe L.

Dans ce paragraphe, on explique comment décider le gagnant depuis une position de la forme $(p_{in},\bot)$ dans $\mathbb{G}$. Pour cela, on considère un STVPA déterministe $\mathcal{A} = \langle S,A,\Sigma,\bot,i,\rho,\delta\rangle$ acceptant le langage L. On a donc un graphe de jeu $\mathcal{G}$ sur un VPP, dont les arcs sont étiquetés par un alphabet d'actions A. La condition de gain est un langage ω-VPL sur l'alphabet A, qui est décrit par un STVPA déterministe $\mathcal{A}$. On se retrouve à peu de choses près dans le cadre du paragraphe 6.3.1, où l'on avait un graphe de jeu fini sur lequel on considérait une condition de gain décrite par un automate à pile déterministe de parité sur l'alphabet d'étiquetage. On avait alors considéré le produit entre le graphe de jeu et l'automate à pile déterministe de parité, et obtenu un jeu de parité sur un processus à pile qui permettait de décider le gagnant dans le premier jeu.

On raisonne de la même façon ici. Comme $\mathcal{A}$ est déterministe et fonctionne sur le même alphabet d'entrée que $\mathcal{P}$, on va considérer le produit $\mathcal{P} \times \mathcal{A}$. Ce dernier va être un STVPA, qui engendre un jeu de parité en escalier $\mathbb{G}'$ sur un graphe de processus à pile. La partition des états de $\mathcal{P} \times \mathcal{A}$ est induite par celle des états de $\mathcal{P}$, tandis que la fonction de coloriage est induite par celle de $\mathcal{A}$. Il est important de remarquer que le fait que $\mathcal{A}$ soit un STVPA est crucial, puisque dès lors les piles de $\mathcal{A}$ et de $\mathcal{P}$ sont toujours de même hauteur. Cette synchronisation des piles de $\mathcal{A}$ et de $\mathcal{P}$ permet donc de ne pas perdre d'information lorsque l'on prend le produit des piles.

Plus précisément, on considère le VPP $\mathcal{P} \times \mathcal{A} = \langle Q \times S,A,(\Gamma \setminus \{\bot\}) \times (\Sigma \setminus \{\bot\}) \cup \{\bot\},\bot,\Delta'\rangle$ où :

- pour tout $(q,s),(q',s') \in Q \times S$ et tout $(\gamma,\sigma),(\gamma',\sigma') \in (\Gamma \setminus \{\bot\}) \times (\Sigma \setminus \{\bot\})$, on a :
  - pour tout $a \in A_c$, $push((q',s'),(\gamma',\sigma')) \in \Delta'((q,s),(\gamma,\sigma),a)$ si et seulement si $push(q',\gamma') \in \Delta(q,\gamma,a)$ et $\delta(s,\sigma,a) = \{push(s',\sigma')\}$.
  - pour tout $a \in A_{int}$, $skip((q',s')) \in \Delta'((q,s),(\gamma,\sigma),a)$ si et seulement si $skip(q') \in \Delta(q,\gamma,a)$ et $\delta(s,\sigma,a) = \{skip(s')\}$.
  - pour tout $a \in A_r$, $pop((q',s')) \in \Delta'((q,s),(\gamma,\sigma),a)$ si et seulement si $pop(q') \in \Delta(q,\gamma,a)$ et $\delta(s,\sigma,a) = \{pop(s')\}$.
- pour tout $(q,s),(q',s') \in Q \times S$ et tout $(\gamma',\sigma') \in (\Gamma \setminus \{\bot\}) \times (\Sigma \setminus \{\bot\})$, on a :
  - pour tout $a \in A_c$, $push((q',s'),(\gamma',\sigma')) \in \Delta'((q,s),\bot,a)$ si et seulement si $push(q',\gamma') \in \Delta(q,\bot,a)$ et $\delta(s,\bot,a) = \{push(s',\sigma')\}$.
  - pour tout $a \in A_{int} \cup A_r$, $skip((q',s')) \in \Delta'((q,s),\bot,a)$ si et seulement si $skip(q') \in \Delta(q,\bot,a)$ et $\delta(s,\bot,a) = \{skip(s')\}$.

On considère la partition $Q_{\mathbf{E}} \times S \cup Q_{\mathbf{A}} \times S$ des états de $\mathcal{P} \times \mathcal{A}$ et la fonction de coloriage ρ' sur $Q \times S$ en posant $\rho'(q,s) = \rho(s)$. Enfin, on appelle $\mathcal{G}'$ le graphe de jeu

qu'ils engendrent, et $\mathbb{G}'$ le jeu de parité en escalier sur $\mathcal{G}'$. En utilisant les mêmes techniques que pour le théorème 6.5, on prouve le résultat suivant:

Théorème 8.3 *[56] Pour tout état $q_{in} \in Q$, la configuration $(q_{in},\bot)$ est gagnante pour Eve dans le jeu $\mathbb{G}$ si et seulement si la configuration $((q_{in},s_{in}),\bot)$ est gagnante pour Eve dans le jeu $\mathbb{G}'$.*

Remarque 8.1 Comme dans le cas du théorème 6.5, on peut à partir d'une stratégie gagnante dans $\mathbb{G}'$ déduire une stratégie gagnante dans $\mathbb{G}$. Cette dernière ne nécessite de plus qu'une quantité finie de mémoire supplémentaire.

8.1.4 Complexité

Bornes supérieures

Les théorèmes 8.1, 8.3 et 5.6 impliquent le corollaire suivant.

Corollaire 8.1 *[56] Soit $\mathbb{G}$ un jeu sur un graphe de* Vpp *muni d'une condition externe Ω qui est un langage ω-*Vpl*. Pour toute configuration de pile vide $(q_{in},\bot)$, on peut décider si Eve possède une stratégie gagnante dans $\mathbb{G}$ depuis $(q_{in},\bot)$ en* Dexptime *si Ω est décrite par un* StVpa *déterministe ou par un* Vpa *déterministe de Büchi, et en 2-*Dexptime *si Ω est décrite par un* Vpa *non déterministe de Büchi.*

Par ailleurs, en utilisant les résultats donnés dans [1], si la condition de gain est décrite par une formule ψ de Caret, on peut construire un Vpa de Büchi non déterministe qui reconnaît exactement les modèles de ψ. De plus, un tel Vpa peut être choisi de taille $2^{\mathcal{O}(|\psi|)}$. Dès lors, on a le corollaire suivant.

Corollaire 8.2 *[56] Soit $\mathbb{G}$ un jeu sur un graphe de* Vpp *muni d'une condition externe Ω décrite par une formule de* Caret*. Pour toute configuration de pile vide $(q_{in},\bot)$, on peut décider si Eve possède une stratégie gagnante dans $\mathbb{G}$ depuis $(q_{in},\bot)$ en 3-*Dexptime*.*

En particulier, comme Caret est une extension de la logique ltl, on a :

Corollaire 8.3 *[56] Soit $\mathbb{G}$ un jeu sur un graphe de* Vpp *muni d'une condition externe Ω décrite par une formule de* ltl*. Pour toute configuration de pile vide $(q_{in},\bot)$, on peut décider si Eve possède une stratégie gagnante dans $\mathbb{G}$ depuis $(q_{in},\bot)$ en 3-*Dexptime*.*

Bornes inférieures

Concernant les bornes inférieures, on a établi dans [56] le résultat suivant.

Théorème 8.4 *[56] Soit $\mathbb{G}$ un jeu sur un graphe de* Vpp *muni d'une condition externe Ω décrite par une formule de* ltl*. Décider si Eve possède une stratégie gagnante dans $\mathbb{G}$ depuis une configuration de pile vide est un problème 3-*Dexptime*-dur.*

Preuve. La preuve est technique. On montre comment simuler une machine de Turing alternante d'espace doublement exponentiel.

Comme dans la preuve du théorème 6.8, on va coder une configuration d'une machine de Turing à l'aide de compteurs imbriqués. On considère une machine de Turing alternante qui, sur une entrée de taille n, utilise un espace doublement exponentiel en m, où m est polynomial en n.

Une configuration $a_0a_1\cdots a_{2^{2^m}-1}$ est codée par $a_0n_0a_1n_1\cdots a_{2^{2^m}-1}n_{2^{2^m}-1}$, où $n_i = n_{i,0}(n_{i,0}^0n_{i,0}^1\cdots n_{i,0}^{m-1})n_{i,1}(n_{i,1}^0n_{i,1}^1\cdots n_{i,1}^{m-1})\cdots n_{i,2^m-1}(n_{i,2^m-1}^0n_{i,2^m-1}^1\cdots n_{i,2^m-1}^{m-1})$, $n_{i,0}n_{i,1}\cdots n_{i,2^m-1}$ est la représentation binaire de i, et $n_{i,j}^0n_{i,j}^1\cdots n_{i,j}^{m-1}$ est la représentation binaire de j. Dans une telle description, les bits de la forme $n_{i,j}^k$ sont qualifiés de bit de niveau inférieur, tandis que les bits de la forme $n_{i,j}$ sont qualifiés de bit de niveau supérieur. De même, on qualifie $n_{i,j}^0n_{i,j}^1\cdots n_{i,j}^{m-1}$ de compteur de niveau inférieur, et $n_{i,0}n_{i,1}\cdots n_{i,2^m-1}$ de compteur de niveau supérieur.

Le jeu se passe de la façon suivante : Eve décrit la configuration initiale puis Eve (si la configuration est existentielle) ou Adam (sinon) choisit une transition de la machine de Turing. Ensuite, Eve décrit la configuration successeur et ainsi de suite. La description est faite en empilant des lettres dans la pile et les transitions sont toujours étiquetées par la lettre que l'on empile (et plus tard que l'on dépile).

Eve remporte une partie si une configuration finale est décrite. Il n'est pas difficile de voir que l'on peut coder cette condition à l'aide d'une formule LTL.

Bien sûr, Adam peut contester les configurations décrites par Eve, ce qui permet d'éviter qu'elle ne triche. Pour cela, il peut après toute description d'une configuration marquer un symbole spécial $*$ et dépiler tant qu'il veut en intercalant à deux reprises un marqueur obj_1 ou obj_2. Dans ce cas, la suite des étiquetages est de la forme $\sigma * \sigma'$ où σ' est un préfixe du miroir de σ (on a dépilé donc lu à l'envers) dans lequel deux marqueurs sont intercalés. Les différentes vérifications sont les suivantes :

- la consécution des compteurs de niveau inférieur est facilement vérifiée par une formule LTL. En effet, on peut spécifier la valeur du j-ème bit d'un compteur de niveau inférieur en fonction des m bits du compteur précédent.
- si Adam pense qu'il y a un problème sur un bit de niveau supérieur, il dépile jusqu'à trouver le premier bit à problème, qu'il marque en faisant une transition *skip* d'étiquette obj_1 puis il continue de dépiler jusqu'au bit supérieur qui n'a pas été correctement mis à jour, qu'il marque également avec obj_1. Une formule LTL permet de vérifier si le premier bit a mal été actualisé. Cependant, pour éviter qu'Adam ne triche en indiquant des bits d'indices différent, une formule LTL vérifie également que les compteurs de niveau inférieurs sont les mêmes (ce qui se fait sans difficulté).
- si Adam pense qu'il y a un problème sur une lettre ou sur l'état de la configuration, il dépile les deux dernières configurations et marque dans chacune, à l'aide d'une transition *skip* étiquetée par un symbole obj_2, la lettre (ou l'état) posant problème. On vérifie avec une formule LTL si l'actualisation a été correcte ou non. Il faut également faire en sorte qu'Adam ne puisse pas tricher en choisissant des cases qui ne se correspondent pas. Pour cela, on autorise Eve a utiliser, une fois qu'Adam a écrit les deux marqueurs obj_2, un marqueur *notok* pour signaler un bit faux dans le compteur de niveau supérieur associée à la case incriminée. Pour vérifier alors cette objection d'Eve, on utilise

une formule LTL qui vérifie si, dans le premier compteur d'ordre supérieur, le bit incriminé par Eve est bien le même que dans le second. Pour cela, la formule teste de façon exhaustive (à l'aide d'une conjonction d'implications) les compteurs d'ordre inférieur jusqu'à trouver le bon.

Suite à une contestation, la partie se termine et le gagnant est celui qui avait raison au sujet de cette dernière.

Le caractère polynomial et l'existence de telles formules LTL peuvent être établis en détail mais la preuve est fastidieuse et nécessite un haut niveau de détail. Pour plus de détails, on renvoie le lecteur à [56]. Le fait qu'Eve possède une stratégie gagnante si et seulement si la machine de Turing accepte la configuration initiale ne pose quant à lui pas de problème. □

Signalons une conséquence du résultat précédent.

Corollaire 8.4 *[56] Soit $\mathbb{G}$ un jeu sur un graphe de processus à pile muni d'une condition externe Ω décrite par un automate non déterministe de Büchi. Décider si Eve possède une stratégie gagnante dans $\mathbb{G}$ depuis une configuration de pile vide est un problème 2-*DEXPTIME*-dur.*

Preuve. On simule cette fois une machine de Turing alternante d'espace exponentiel. On considère donc une telle machine avec une entrée de taille n. En posant $n' = \log n$, et en utilisant la réduction précédente, on construit une formule de LTL ψ, polynomiale en n', et un jeu équivalent sur un processus à pile de taille polynomiale en n. La formule ψ peut alors être transformée en un automate non déterministe de Büchi acceptant les modèles de ψ. De plus, un tel automate peut être choisi de taille exponentielle en n', c'est-à-dire polynomial en n. On obtient ainsi un jeu sur un graphe de processus à pile de taille polynomiale en n, et muni d'une condition de gain décrite par un automate non déterministe de Büchi de taille également polynomiale en n. Par ailleurs, Eve possède une stratégie gagnante depuis une configuration de pile vide particulière dans ce jeu si et seulement si la machine de Turing précédente accepte sur son entrée initiale. On a donc bien donné une réduction polynomiale qui prouve le résultat voulu. □

Comme la logique LTL est une sous-classe de la logique CARET, et que les automates de Büchi non déterministes sont un cas particulier de VPA non déterministes de Büchi, on a le résultat suivant.

Corollaire 8.5 *[56] Décider le gagnant dans un jeu sur un graphe de* VPP *muni d'une condition ω-VPL décrite par un* VPA *non déterministe de Büchi est un problème 2-*DEXPTIME*-complet.*

Décider le gagnant dans un jeu sur un graphe de processus à pile muni d'une condition décrite par une formule LTL *est un problème 3-*DEXPTIME*-complet.*

Décider le gagnant dans un jeu sur un graphe de VPP *muni d'une condition décrite par une formule* CARET *est un problème 3-*DEXPTIME*-complet.*

8.1.5 Stratégies

Dans la proposition 5.5, nous avons montré que dans un jeu de parité en escalier sur un graphe de processus à pile, une mémoire infinie peut être nécessaire dans une

stratégie gagnante. Une adaptation simple de la preuve permet d'établir le résultat suivant.

Proposition 8.1 *[56] Il existe un jeu sur un graphe de* VPP *muni d'une condition de gain ω-*VPL *et une position gagnante pour Eve dans ce jeu tels que toute stratégie gagnante pour Eve nécessite une mémoire infinie.*

Par ailleurs, nous avons vu dans le chapitre 5 que l'on peut construire des stratégies à pile pour Eve dans un jeu de parité en escalier sur un graphe de processus à pile. Dès lors, en utilisant la remarque 8.1, on déduit le résultat suivant.

Proposition 8.2 *[56] Dans un jeu sur un graphe de* VPP *muni d'une condition de gain ω-*VPL*, Eve possède depuis toute position gagnante une stratégie à pile.*

8.1.6 Complexité topologique

On donne maintenant la complexité topologique des langages ω-VPL, et par là-même celle des conditions de gains externes que nous venons d'étudier.

Théorème 8.5 *[56] La classe des langages ω-*VPL *est incluse dans $\mathcal{B}(\Sigma_3)$.*

Preuve. Soit L un langage ω-VPL sur un alphabet d'actions A, et soit $\mathcal{A}$ un STVPA qui reconnaît L. Pour tout état q de $\mathcal{A}$, on définit le langage L_q des mots infinis u pour lesquels $\mathcal{A}$ visite infiniment souvent l'état q dans des positions de $Steps_u$. On va montrer que tout langage L_q est dans Π_3. Comme le langage L s'exprime ensuite facilement comme une combinaison booléenne des langages L_q, on conclut sans problème la preuve.

On appelle U_q l'ensemble des mots finis qui conduisent $\mathcal{A}$ dans l'état q. Le langage L_q s'exprime par $L_q = \bigcap_{m\in\mathbb{N}} \bigcup_{n>m} L_{q,n}$, où $L_{q,n}$ désigne le langage des mots infinis u tels $n \in Steps_u$ et $u{\restriction_n} \in U_q$. On va montrer que tout langage $L_{q,n}$ est fermé, ce qui implique que L_q est Π_3 et termine la preuve.

Etant donné un mot infini u et un indice $n \geq 0$, on a $n \in Steps_u$ dans deux cas :

- le mot $u{\restriction_n}$ est de hauteur de pile nulle, c'est-à-dire que $sh(u{\restriction_n}) = 0$. On appelle $U_0 = (A_r \cup A_{int} \cup L_{wmw})^*$ l'ensemble des mots finis de hauteur de pile nulle.
- dans le suffixe de u commençant à la position d'indice n, tout retour est associé à un appel. On appelle $U_{mr} = (A_c \cup A_{int} \cup L_{wmw})^*$ l'ensemble des mots dans lesquels tout retour est associé à un appel.

On a alors

$$L_{q,m} = [(U_q \cap A^n)A^\omega] \cap \left[(U_0 \cap A^n)A^\omega \cup \left(\bigcap_{n'>n} (A^n U_{mr} \cap A^{n'})A^\omega\right)\right]$$

Tous les langages de base intervenant dans la formule précédente sont de la forme KA^ω, où K est un langage fini. Ce sont donc des ouverts fermés, et dès lors $L_{q,n}$ est fermé. □

8.2 Combinaisons booléennes d'un condition de parité et d'une condition sur la hauteur de pile

On étudie maintenant la résolution d'un jeu sur un graphe de processus à pile muni d'une condition de gain qui est une combinaison booléenne d'une condition de parité et d'une condition sur la hauteur de pile. Une telle condition est donc de la forme parité et/ou explosion/bornage/explosion stricte/répétition.

Toutes ces conditions de gain étant invariantes par translations horizontales et verticales, les résultats du chapitre 4 s'appliquent, et l'on peut se restreindre au calcul des ensembles de retours et des configurations gagnantes de pile vide.

Savoir résoudre ce genre de jeux est particulièrement intéressant pour la vérification. En effet, cela donne une méthode pour synthétiser un contrôleur pour qu'un système modélisé par un graphe de processus à compteur vérifie une spécification demandant que le nombre d'appel récursif soient bornés (condition de bornage) et qu'en même temps une condition régulière, données par exemple par une formule de LTL ou CTL, soit satisfaite.

8.2.1 Présentation des constructions

Considérons le jeu suivant, où il n'y a qu'un seul joueur, Eve. Le jeu se déroule sur un graphe de processus à pile, et il est équipé d'une condition de type Büchi et explosion. Eve peut empiler ou dépiler comme elle veut, mais ne peut atteindre l'état final que si la pile est vide. Ainsi, pour gagner, Eve est forcée d'empiler des symboles (afin d'exploser la pile), puis de les dépiler (afin de passer par l'état final) et ainsi de suite. Une mémoire infinie est alors nécessaire pour se souvenir de la plus grande hauteur de pile atteinte. On a donc le résultat suivant.

Proposition 8.3 *Il existe un jeu sur un graphe de processus à pile muni d'une condition de type Büchi et explosion, et une configuration gagnante pour Eve tels que toute stratégie gagnante pour Eve depuis cette position nécessite une mémoire infinie.*

Dans le chapitre 5, nous avons montré que la condition d'explosion et la condition d'explosion stricte étaient les mêmes. L'exemple précédent montre que ce n'est plus le cas lorsqu'on les combine avec des conditions de parité. En effet, dans le jeu précédent, Eve possède une stratégie gagnante pour la condition de parité et explosion, mais pas pour la condition de parité et explosion stricte.

Pour résoudre de tels jeux, on commence dans un premier temps à raffiner la notion de M/B factorisation en disant que la hauteur d'une bosse est l'augmentation maximale de la hauteur de pile au cours de celle-ci. On a alors le résultat suivant qui raffine la proposition 5.1

Proposition 8.4 *Soit une partie Λ dans un jeu sur un graphe d'automate à pile. On a alors :*

- *la pile dans Λ explose si et seulement si la M/B factorisation de Λ contient une infinité de marches ou contient des bosses arbitrairement hautes.*

- *la pile dans Λ est bornée si et seulement si la M/B factorisation de Λ ne contient qu'un nombre fini de marches et si la hauteur des bosses est bornée.*
- *la pile dans Λ explose strictement si et seulement si la M/B factorisation de Λ contient une infinité de marches.*
- *une hauteur de pile est infiniment souvent visitée dans Λ si et seulement si la M/B factorisation de Λ contient un nombre fini de marches.*

Reprenons maintenant le jeu informel entre Eve et Adam. Dans le cas des conditions de parité, Eve annonçait quels états pouvaient être atteints lors d'un dépilement en fonction de la plus petite couleur visitée depuis l'empilement. Maintenant, on souhaite en plus avoir des précisions sur la hauteur des bosses. Cette dernière pouvant être arbitrairement grande, on va se contenter de la comparer à un seuil κ. Lorsque l'on aura fait une bosse de hauteur plus grande que κ, on incrémentera le seuil.

L'objet alors obtenu n'est plus un graphe de jeu fini, mais un graphe de jeu sur un processus à compteur, le compteur codant le seuil.

Dans tout ce qui suit, on considère un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$. On appelle $G = (V,E)$ le graphe qu'il engendre, on se donne une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de Q, et l'on considère le graphe de jeu $\mathcal{G} = (G,V_{\mathbf{E}},V_{\mathbf{A}})$ associé à G et à cette partition de Q. Enfin, on se donne une fonction de coloriage ρ de Q dans un ensemble fini $\{0,\ldots,d\}$ de couleurs.

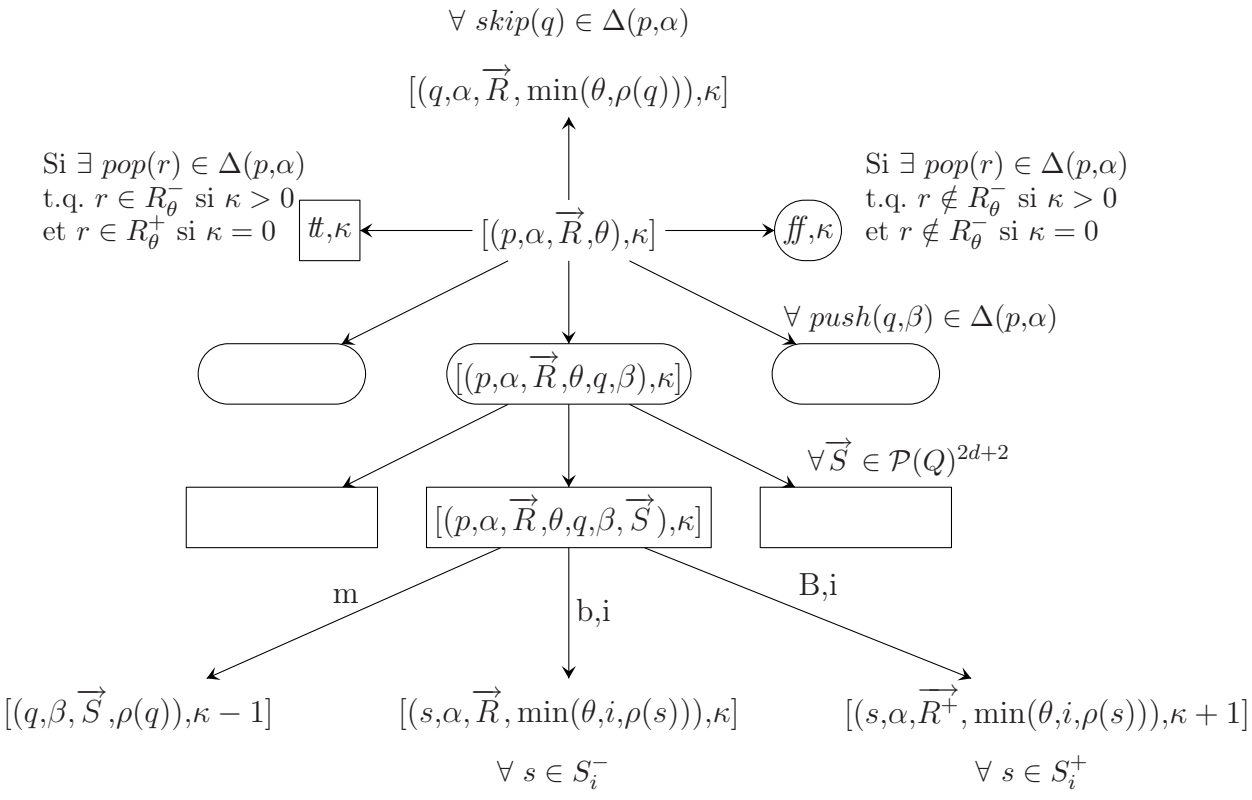


FIG. 8.2 – *Structure locale de $\widetilde{\mathcal{G}}$.*

On considère alors le graphe de jeu $\widetilde{\mathcal{G}}$ sur un processus à compteur, illustré dans la figure 8.2. Pour une valeur κ du compteur, on désigne par $\kappa-1$ la valeur obtenue en décrémentant le compteur (c'est-à-dire en dépilant le sommet), avec la convention que si $\kappa=0$, $\kappa-1=0$. Le graphe $\mathcal{G}$ est construit comme suit.

- les sommets principaux de $\widetilde{\mathcal{G}}$ sont ceux de la forme $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$, où $p\in Q$, $\alpha\in\Gamma$, $\overrightarrow{R}=((R_0^-,R_0^+)\ldots,(R_d^-,R_d^+))\in\mathcal{P}(Q)^{2d+2}$, $\theta\in\{0,\ldots,d\}$ et $\kappa\geq 0$. Un sommet $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ représente une partie partielle Λ dans $\mathbb{G}$ telle que :
  - le dernier sommet de Λ est de la forme $(p,\alpha\sigma)$ pour un certain $\sigma\in\Gamma^*$.
  - Eve prétend pouvoir jouer depuis Λ de telle sorte que si α est un jour dépilé, l'état de contrôle atteint après avoir dépilé α est dans $R_m^\star$, où m est la plus petite couleur vue lorsque α se trouvait dans la pile, et $\star=+$ si l'augmentation maximale de la hauteur de pile a été d'au moins κ lorsque α se trouvait dans la pile, et $\star=-$ sinon.
  - la plus petite couleur vue depuis le moment où α a été empilé est θ.

  Un sommet de la forme $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ appartient à Eve si et seulement si $p\in Q_{\mathbf{E}}$.
- les états tt et ff sont là pour s'assurer que les vecteurs $\overrightarrow{R}$ dans les sommets principaux sont corrects. Le sommet $[\mathit{tt},\kappa]$ appartient à Adam, tandis que $[\mathit{ff},\kappa]$ appartient à Eve. Ces sommets étant des culs-de-sac, une partie qui arrive dans $[\mathit{tt},\kappa]$ sera remportée par Eve, tandis qu'une partie arrivant dans $[\mathit{ff},\kappa]$ sera reportée par Adam.

  Il y a une transition d'un sommet $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ vers $[\mathit{tt},\kappa]$, si et seulement s'il existe une règle de la forme $pop(r)\in\Delta(p,\alpha)$, telle que $r\in R_\theta^\star$ avec $\star=-$ si $\kappa>0$ et $\star=+$ sinon (ce qui traduit le fait que $\overrightarrow{R}$ est correct par rapport à cette règle). Symétriquement, il y a une transition d'un sommet $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ vers $[\mathit{ff},\kappa]$ si et seulement s'il existe une règle de la forme $pop(r)\in\Delta(p,\alpha)$, telle que $r\notin R_\theta^\star$ avec $\star=-$ si $\kappa>0$ et $\star=+$ sinon (ce qui traduit le fait que $\overrightarrow{R}$ n'est pas correct par rapport à cette règle).
- afin de simuler l'application d'une transition $skip(q)\in\Delta(p,\alpha)$, le joueur à qui appartient $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ va en $[(q,\alpha,\overrightarrow{R},\min(\theta,\rho(q))),\kappa]$. Remarquons que dans ce cas la dernière composante doit être actualisée, puisque la plus petite couleur vue depuis que α est dans la pile est désormais $\min(\theta,\rho(q))$.
- afin de simuler l'application d'une transition $push(q,\beta)\in\Delta(p,\alpha)$, le joueur à qui appartient $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ va en $[(p,\alpha,\overrightarrow{R},\theta,q,\beta),\kappa]$. Ce dernier sommet appartient à Eve et celle-ci doit donc choisir un vecteur $\overrightarrow{S}=((S_0^-,S_0^+),\ldots,(S_d^-,S_d^+))\in\mathcal{P}(Q)^{2d+2}$ décrivant les états pouvant être atteints si β est un jour dépilé. Eve va alors dans le sommet $[(p,\alpha,\overrightarrow{R},\theta,q,\beta,\overrightarrow{S}),\kappa]$.

  Ce dernier appartient à Adam. Depuis $[(p,\alpha,\overrightarrow{R},\theta,q,\beta,\overrightarrow{S}),\kappa]$, Adam choisit s'il veut simuler une bosse ou une marche. Dans le premier cas, il a le choix de simuler une bosse de hauteur (strictement) inférieure à κ ou supérieure à κ. Pour simuler une bosse de hauteur inférieure à κ, il va dans un sommet $[(s,\alpha,\overrightarrow{R},\min(\theta,i,\rho(s))),\kappa]$, pour un certain $i\in\{0,\ldots,d\}$ et un $s\in S_i^-$, via un arc bicolore, de couleurs i et b (b signifiant petite bosse).

Pour simuler une bosse de hauteur supérieure à κ, il va dans un sommet $[(s,\alpha,\overrightarrow{R^+},\min(\theta,i,\rho(s))),\kappa+1]$, pour un certain $i \in \{0,\ldots,d\}$ et un $s \in S_i^+$, via un arc bicolore, de couleurs i et B (B signifiant grande bosse), et où $\overrightarrow{R^+} = ((R_0^+,R_0^+),\ldots,(R_d^+,R_d^+))$.

Enfin, pour simuler une marche, Adam va dans le sommet $[(q,\beta,\overrightarrow{S},\rho(q)),\kappa-1]$, via un arc coloré par m.

Dans le cas des bosses, la dernière composante est mise à jour : on vient de faire une bosse de couleur i et donc la plus petite couleur visitée depuis que α a été empilé est désormais $\min(\theta,i,\rho(s))$. Dans le cas d'une marche, on initialise la dernière composante, la seule couleur vue depuis que β a été empilé étant $\rho(q)$. Dans le cas d'une bosse de hauteur supérieure à κ, on met également à jour le vecteur d'ensembles d'états : on vient de faire une grande bosse et si α est un jour dépilé, l'augmentation maximale de la pile aura donc été d'au moins κ. Dès lors, l'état atteint devra être dans un R_m^+ pour une certaine couleur m.

Le graphe de jeu $\widetilde{\mathcal{G}}$ est muni d'un coloriage (partiel) sur les sommets : les sommets colorés sont ceux de la forme $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ et leur couleur est $\rho(p)$. On a aussi un multi-coloriage partiel sur les arcs partant des sommets de la forme $[(p,\alpha,\overrightarrow{R},\theta,q,\beta,\overrightarrow{S}),\kappa]$: ce multicoloriage est toujours formé d'une couleur dans $\{m,b,B\}$ complétée éventuellement d'une couleur dans $\{0,1,\ldots,d\}$.

Remarque 8.2 On peut objecter que le jeu $\widetilde{\mathbb{G}}$ n'est pas exactement conforme à la définition classique d'un jeu de parité, puisque $\widetilde{\mathcal{G}}$ n'est que partiellement coloré, et que l'on autorise un multicoloriage sur les arcs. Cependant, on définit sans grande difficulté un graphe de jeu équivalent, dans lequel seul les sommets sont colorés.

8.2.2 Condition de parité et d'explosion

On appelle $\mathbb{G}$ le jeu sur $\mathcal{G}$ muni de la condition de gain parité et explosion.

On appelle $\widetilde{\mathbb{G}}$ le jeu sur $\widetilde{\mathcal{G}}$ équipé de la condition de gain suivante : Eve gagne une partie Λ si et seulement si Λ vérifie la condition de parité (pour $\{0,\ldots,d\}$) et si elle visite infiniment souvent l'ensemble de couleurs $\{m,B\}$. On a alors le résultat suivant.

Théorème 8.6 *Soit $p_{in} \in Q$, soit $\alpha_{in} \in \Gamma$ et soit $T \subseteq Q$. On a les équivalences suivantes :*

- *Eve possède une stratégie gagnante dans $\mathbb{G}(T)$ depuis $(p_{in},\alpha_{in}\bot)$ si et seulement si elle possède une stratégie gagnante dans $\widetilde{\mathbb{G}}$ depuis $[(p_{in},\alpha_{in},(T,\ldots,T),\rho(p_{in})),0]$.*
- *Eve possède une stratégie gagnante dans $\mathbb{G}$ depuis $(p_{in},\bot)$ si et seulement si elle possède une stratégie gagnante dans $\widetilde{\mathbb{G}}$ depuis $[(p_{in},\bot,(\emptyset,\ldots,\emptyset),\rho(p_{in})),0]$.*

Les preuves de ces deux points étant similaires, nous nous contenterons de celle du premier.

8.2.3 Preuve du théorème 8.6

On commence par introduire la notion de M/B factorisation pour les parties dans le jeu $\widetilde{\mathbb{G}}$. On montre ensuite les deux implications.

Factorisation dans $\widetilde{\mathbb{G}}$

Etant donné que $\widetilde{\mathcal{G}}$ est muni d'un coloriage sur les arcs, une partie dans $\widetilde{\mathbb{G}}$ devrait être représentée comme un couple formé d'un sommet initial et d'une suite d'arcs. Cependant, pour alléger les notations, on représente une partie comme une suite de sommets, dans laquelle on intercale, entre deux sommets reliés par un arc coloré, la couleur de l'arc. Les couleurs que nous intercalerons seront prises dans $\{0,\dots,d\}$ car pour ce qui est des couleurs $\{m,b,B\}$, leur présence se détecte automatiquement grâce à une augmentation de la pile (pour B) et à une non-augmentation de la pile en présence d'une couleur dans $\{0,\dots,d\}$ (pour b).

Remarquons tout d'abord qu'une partie dans $\widetilde{\mathbb{G}}$ visite *régulièrement*, au plus tous les trois coups, des sommets de la forme $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$. On qualifie de *round* la séquence de sommets entre deux passages dans de telles positions. Un round peut alors être de plusieurs types :

- il est de la forme $[(p,\alpha,\overrightarrow{R},\theta),\kappa][(q,\alpha,\overrightarrow{R},\theta),\kappa]$, et correspond donc à la simulation d'une règle de type skip. On parle alors de *bosse de hauteur nulle*.
- il est de la forme $[(p,\alpha,\overrightarrow{R},\theta),\kappa][(p,\alpha,\overrightarrow{R},\theta,q,\beta),\kappa][(p,\alpha,\overrightarrow{R},\theta,q,\beta,\overrightarrow{S}),\kappa]i$ $[(s,\alpha,\overrightarrow{R},\min(\theta,i,\rho(s))),\kappa]$, et correspond donc à la simulation d'une règle empilant un symbole β suivie d'une série de coups se terminant par le dépilement de β, et au cours desquels la pile augmente de moins de κ symboles. On le qualifie de *bosse de hauteur inférieure à κ*.
- il est de la forme $[(p,\alpha,\overrightarrow{R},\theta),\kappa][(p,\alpha,\overrightarrow{R},\theta,q,\beta),\kappa][(p,\alpha,\overrightarrow{R},\theta,q,\beta,\overrightarrow{S}),\kappa]i$ $[(s,\alpha,\overrightarrow{R^+},\min(\theta,i,\rho(s))),\kappa+1]$, et correspond donc à la simulation d'une règle empilant un symbole β suivie d'une série de coups se terminant par le dépilement de β, et au cours desquels la pile augmente d'au moins κ symboles. On le qualifie de *bosse de hauteur supérieure à κ*.
- il est de la forme $[(p,\alpha,\overrightarrow{R},\theta),\kappa][(p,\alpha,\overrightarrow{R},\theta,q,\beta),\kappa][(p,\alpha,\overrightarrow{R},\theta,q,\beta,\overrightarrow{S}),\kappa]$ $[(q,\beta,\overrightarrow{S},\rho(q)),\kappa-1]$ et correspond donc à la simulation d'une règle d'empilement d'un symbole β qui ne sera pas dépilé. On le qualifie de *marche*.

Etant donnée une partie $\lambda = v_0v_1v_2\cdots$ dans $\widetilde{\mathbb{G}}$, on peut considérer le sous-ensemble d'indices correspondant à des sommets de la forme $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$. Plus précisément :

$$Steps_\lambda = \{n \in \mathbb{N} \mid v_n = [(p,\alpha,\overrightarrow{R},\theta),\kappa], p \in Q, \alpha \in \Gamma, \overrightarrow{R} \in \mathcal{P}(Q)^{2d+2}, 0 \leq \theta \leq d, \kappa \geq 0\}$$

Tout comme dans le cas des jeux sur un graphe de processus à pile, l'ensemble $Steps_\lambda$ induit une factorisation naturelle d'une partie λ en marches et en bosses.

Définition 8.2 (M/B factorisation) *Etant donnée une partie, partielle ou non, $\lambda = v_0v_1v_2\cdots$, on appelle* Marche/Bosse factorisation *de λ, ou plus simplement*

M/B factorisation*, la suite, finie or infinie,* $(\lambda_i)_{i\geq 0}$ *de rounds de* λ *définis de la façon suivante. Soit* $Steps_\lambda = \{n_0 < n_1 < n_2 < \cdots\}$*, on pose alors, pour tout* $0 \leq i < |Steps_\lambda|$, $\lambda_i = v_{n_i} \cdots v_{n_{i+1}}$.

Ainsi, pour tout $i \geq 0$*, le premier sommet de* λ_{i+1} *est le dernier sommet de* λ_i*. De plus,* $\lambda = \lambda_1 \odot \lambda_2 \odot \lambda_3 \odot \cdots$*, où* $\lambda_i \odot \lambda_{i+1}$ *désigne la concaténation de* λ_i *et de* λ_{i+1} *privé de son premier sommet.*

Enfin, la couleur *d'un facteur désigne la plus petite couleur des sommets et arcs qui le composent.*

Afin de prouver les deux implications du théorème 8.6, on construira, à partir d'une stratégie gagnante pour Eve dans l'un des deux jeux, une stratégie dans l'autre jeu. Cette stratégie utilise une mémoire qui code une partie dans le premier jeu, où Eve respecte sa stratégie gagnante, et qui est donc une partie gagnante. L'argument principal pour prouver que la nouvelle stratégie est gagnante consiste essentiellement à montrer une correspondance entre les M/B factorisations des deux parties et à conclure en utilisant le fait que la première partie est gagnante.

Implication directe

Supposons que la configuration $(p_{in},\alpha_{in}\bot)$ soit gagnante dans le jeu $\mathbb{G}(T)$, et considérons une stratégie Φ gagnante pour Eve dans $\mathbb{G}(T)$ depuis $(p_{in},\alpha_{in}\bot)$.

A partir de Φ, on définit une stratégie φ pour Eve dans $\widetilde{\mathbb{G}}$ depuis $[(p_{in},\alpha_{in},(T,\ldots,T),\rho(p)),0]$. La stratégie utilise une mémoire qui contient une partie partielle dans $\mathbb{G}$, c'est-à-dire un élément de V^*, et son contenu sera noté Λ. Au départ, Λ est réduit au sommet $(p_{in},\alpha_{in}\bot)$. On commence par décrire φ, puis on explique comment Λ est mis à jour. La stratégie φ, ainsi que la mise à jour de Λ, sont décrites pour un round.

Choix du coup : On suppose donc que l'on est dans une configuration de la forme $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ avec $p \in Q_{\mathbf{E}}$. Le coup donné par φ dépend alors de $\Phi(\Lambda)$:

- si $\Phi(\Lambda) = pop(r)$, alors Eve va dans $[\sharp,\kappa]$ (la proposition 8.5 montrera que ce coup est toujours possible).
- si $\Phi(\Lambda) = skip(q)$, alors Eve va dans $[(q,\alpha,\overrightarrow{R},\min(\theta,\rho(q))),\kappa]$.
- si $\Phi(\Lambda) = push(q,\beta)$, alors Eve va dans $[(p,\alpha,\overrightarrow{R},\theta,q,\beta),\kappa]$.

Dans ce dernier cas, ou lorsque $p \in Q_{\mathbf{A}}$ et qu'Adam est allé en $[(p,\alpha,\overrightarrow{R},\theta,q,\beta),\kappa]$, nous devons également expliquer comment Eve joue depuis $[(p,\alpha,\overrightarrow{R},\theta,q,\beta),\kappa]$. Elle doit choisir un vecteur $\overrightarrow{S} \in \mathcal{P}(Q)^{2d+2}$ qui décrit les états atteignables si β est dépilé en fonction de la plus petite couleur visitée entre temps et de l'augmentation de la hauteur de pile par rapport à $\kappa - 1$. Afin de définir $\overrightarrow{S}$, Eve considère l'ensemble des parties possibles dans $\mathbb{G}$, où elle respecte sa stratégie Φ, et qui commencent par $\Lambda \cdot (q,\beta\alpha\sigma)$, si $(p,\alpha\sigma)$ désigne le dernier sommet de Λ. Pour chacune de ces parties, Eve regarde si une configuration de la forme $(s,\alpha\sigma)$ apparaît après $\Lambda\cdot(q,\beta\alpha\sigma)$, c'est-à-dire si β est un jour dépilé. Si tel est le cas, Eve considère la première configuration $(s,\alpha\sigma)$ apparaissant après $\Lambda \cdot (q,\beta\alpha\sigma)$ ainsi que la plus petite couleur visitée i et l'augmentation maximale h de la pile, lorsque β se trouve dans la pile. Pour tout $i \in \{0,\ldots d\}$, S_i^-, est exactement l'ensemble des $s \in Q$, tel que l'on ait la propriété précédente avec $h < \kappa-1$, et S_i^+, est exactement l'ensemble des $s \in Q$, tel que l'on ait

la propriété précédente avec $h \geq \kappa - 1$. Enfin, on pose $\overrightarrow{S} = ((S_0^-,S_0^+),\ldots,(S_d^-,S_d^+))$ et Eve joue alors vers $[(p,\alpha,\overrightarrow{R},\theta,q,\beta,\overrightarrow{S}),\kappa]$.

Mise à jour de Λ : La mise à jour de Λ est effectuée à chaque fois que l'on arrive dans un sommet de la forme $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$. On a alors trois cas, selon la nature du round :

- le round est une bosse de hauteur nulle, et l'on vient donc de simuler une action $skip(q)$. Si l'on appelle $(p,\alpha\sigma)$ le dernier sommet de Λ, on augmente Λ du sommet $(q,\alpha\sigma)$.
- le round est une bosse de hauteur non nulle, et l'on vient donc de simuler une bosse commençant par une action $push(q,\beta)$ et se terminant dans un état s. Si l'on appelle $(p,\alpha\sigma)$ le dernier sommet de Λ, on met à jour Λ en y ajoutant $(q,\beta\alpha\sigma)$ suivi d'une série de coups, où Eve respecte Φ, et qui conduisent à dépiler β et à arriver dans $(s,\alpha\sigma)$ tout en visitant i comme plus petite couleur lorsque β est dans la pile, où i est la couleur de l'arc de la bosse (c'est-à-dire que $s \in S_i^- \cup S_i^+$, si $\overrightarrow{S}$ désigne le vecteur donné par Eve lors du round). De plus, on choisit cette suite de coups de telle sorte à augmenter la pile d'au moins κ symboles, si le round était une bosse de hauteur supérieure à κ, et sinon de moins de κ symboles.
- le round est une marche et l'on vient donc de simuler une action $push(q,\beta)$. Si l'on appelle $(p,\alpha\sigma)$ le dernier sommet de Λ, on augmente Λ du sommet $(q,\beta\alpha\sigma)$.

Dès lors, à toute partie partielle λ dans $\widetilde{\mathbb{G}}$ dans laquelle Eve respecte sa stratégie φ, est associée une partie partielle Λ dans $\mathbb{G}$. Une récurrence immédiate permet de prouver que Λ est une partie dans laquelle Eve respecte Φ. Le même raisonnement peut être mené lorsque λ est une partie infinie, et la partie Λ associée est alors infinie, commence en $(p_{in},\alpha_{in}\bot)$ et lors de celle-ci, Eve respecte sa stratégie gagnante Φ. En particulier, la plus petite couleur infiniment souvent visitée dans Λ est paire et la pile explose.

La proposition suivante découle de la définition de φ

Proposition 8.5 *Soit λ une partie partielle dans $\widetilde{\mathbb{G}}$ commençant en $[(p_{in},\alpha_{in},(T,\ldots,T),\rho(p_{in})),0]$, se terminant dans un sommet de la forme $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$, et où Eve respecte φ. Soit Λ la partie associée à λ construite par la stratégie φ. On a alors les points suivants :*

1. *Λ se termine dans un sommet de la forme $(p,\alpha\sigma)$ pour un certain $\sigma \in \Gamma^*$.*
2. *θ est la plus petite couleur visitée dans Λ depuis que α a été empilé.*
3. *on a visité dans Λ une configuration de hauteur de pile supérieure ou égale à $\kappa + |\sigma|$.*
4. *supposons que la partie Λ soit prolongée en une partie où Eve respecte Φ, et que le coup depuis $(p,\alpha\sigma)$ soit vers une configuration (r,σ). Alors $r \in R_i^\star$, où i est la plus petite couleur visitée lorsque α était dans la pile et $\star = +$ si $\kappa = 0$ et $\star = -$ sinon.*

Preuve. Seul le dernier point est délicat. Il est clair que $r \in R_i^- \cup R_i^+$. On remarque tout d'abord que λ se termine par une succession (qui peut être vide) de bosses de

la forme $[(p',\alpha,\overrightarrow{R'},\theta'),\kappa'] \dots [(p'',\alpha,\overrightarrow{R'^{\star}},\theta''),\kappa'+\iota]$. Appelons $[(p',\alpha,\overrightarrow{R'},\theta'),\kappa']$ le premier sommet de cette série de bosses. Dans le cas particulier où la série est vide, $p'=p$, $\overrightarrow{R'}=\overrightarrow{R}$, $\theta'=\theta$ et $\kappa'=\kappa$. Par ailleurs, on a $\overrightarrow{R}=\overrightarrow{R'}$ et $\kappa'=\kappa$ si toutes les bosses sont de hauteur inférieure à κ', et sinon $\overrightarrow{R}=\overrightarrow{R'^{+}}$ et $\kappa'<\kappa$.

Dans le cas où $\kappa=0$, $\kappa'=0$ et par définition de $R_i'^{+}$, on a $r\in R_i'^{+}=R_i^{+}$. Dans le cas où $\kappa>0$, si $\kappa'=\kappa$, on n'a fait que des bosses inférieures à κ' avant de dépiler α, et donc par définition de $R_i'^{-}$, on a $r\in R_i'^{-}=R_i^{-}$. Si $\kappa'<\kappa$, alors on a fait une bosse de hauteur supérieure à κ' avant de dépiler α, et donc par définition de $R_i'^{+}$, on a $r\in R_i'^{+}=R_i^{-}$. □

Remarque 8.3 La proposition 8.5 implique que la stratégie φ est bien définie lorsqu'elle donne un coup vers $\#$. De même, on en déduit que si Eve respecte φ, le sommet ff ne peut être atteint.

La remarque précédente règle donc le cas des parties finies où Eve respecte φ : elle aboutissent dans le sommet $\#$ et sont remportées par Eve.

Des définitions de $\widetilde{\mathcal{G}}$ et de φ, on déduit la proposition suivante.

Proposition 8.6 *Soit λ une partie infinie dans $\widetilde{\mathbb{G}}$ commençant en $[(p_{in},\alpha_{in},(T,\dots,T),\rho(p_{in})),0]$, où Eve respecte φ. Soit Λ la partie associée à λ construite par la stratégie φ. Soient $(\lambda_i)_{i\geq 1}$ et $(\Lambda_i)_{i\geq 1}$ les M/B factorisations respectives de λ et Λ. On a alors pour tout $i\geq 1$:*

1. *λ_i est un bosse si et seulement si Λ_i est une bosse.*
2. *si λ_i est une bosse de hauteur inférieure à κ, alors Λ_i est une bosse de hauteur inférieure à κ.*
3. *si λ_i est une bosse de hauteur supérieure à κ, alors Λ_i est une bosse de hauteur supérieure à κ.*
4. *λ_i et Λ_i ont même couleur.*

Dès lors, étant donnée une partie infinie λ dans $\widetilde{\mathbb{G}}$ commençant en $[(p_{in},\alpha_{in},(T,\dots,T),\rho(p_{in})),0]$, où Eve respecte φ, l'ensemble des couleurs infiniment souvent répétées dans λ est le même que l'ensemble des couleurs infiniment souvent répétées dans la partie Λ construite par la stratégie φ. Or, Λ est une partie gagnée par Eve, puisque celle-ci respecte Φ. Dès lors, la plus petite couleur infiniment répétée dans λ est paire.

Par ailleurs, la pile dans Λ explose : il y a donc une infinité de marches, ou un niveau est infiniment souvent répété et l'on y effectue des bosses de plus en plus hautes. Dans le premier cas, on en conclut que m est infiniment souvent répété dans λ. Dans le second, on en déduit que B est infiniment souvent répété.

Implication réciproque

On suppose maintenant qu'Eve possède une stratégie gagnante φ dans $\widetilde{\mathbb{G}}$ depuis $[(p_{in},\alpha_{in},(T,\dots,T),\rho(p_{in})),0]$. A partir de φ, on construit une stratégie Φ dans $\mathbb{G}(T)$ depuis $(p_{in},\alpha_{in}\bot)$.

La stratégie Φ est une stratégie utilisant une pile Π, qui stocke dans sa pile de stratégie la description complète d'une partie dans $\widetilde{\mathbb{G}}$. On rappelle qu'une partie

dans $\widetilde{\mathbb{G}}$, c'est-à-dire un chemin dans $\widetilde{\mathcal{G}}$, est vu comme une suite de sommets dans laquelle on intercale des couleurs prises dans $\{0,\dots d\}$.

Ainsi, l'alphabet de pile de Π est l'union des sommets de $\widetilde{\mathcal{G}}$ avec $\{0,\dots,d\}$. Dans la suite, $top(\Pi)$ représente le sommet de la pile de stratégie Π. Par $StCont(\Pi)$, nous désignons le mot obtenu en lisant la pile de stratégie Π de bas en haut (en omettant le symbole de fond de pile). Dans une partie où Eve respecte Φ, $StCont(\Pi)$ sera une partie dans $\widetilde{\mathbb{G}}$ commençant en $[(p_{in},\alpha_{in},(T,\dots,T),\rho(p_{in})),0]$, où Eve respecte sa stratégie gagnante φ. De plus, lors d'une partie Λ où Eve respecte Φ, nous aurons à tout moment que $top(\Pi) = [(p,\alpha,\overrightarrow{R},\theta),\kappa]$ si et seulement si la position courante dans Λ est de la forme $(p,\alpha\sigma)$. Enfin, si Eve respecte Φ, et si α est un jour dépilé, la configuration alors atteinte sera de la forme (r,σ) avec $r \in R_i^\star$, où i est la plus petite couleur visitée lorsque α est dans la pile, et $\star = +$ si la pile augmente d'au moins κ symboles entre $(p,\alpha\sigma)$ et (r,σ), et $\star = -$ sinon.

Afin de décrire Φ, on suppose que l'on se trouve dans une configuration $(p,\alpha\sigma)$ et que $top(\Pi) = [(p,\alpha,\overrightarrow{R},\theta),\kappa]$. Dans un premier temps, nous décrivons comment Eve joue si $p \in Q_{\mathbf{E}}$, et nous expliquons ensuite comment la pile de stratégie est mise à jour.

- **Choix du coup :** Supposons que $p \in Q_{\mathbf{E}}$ et qu'Eve doive donc jouer depuis $(p,\alpha\sigma)$. Pour cela, Eve considère la valeur de φ sur $StCont(\Pi)$.
  S'il s'agit d'un coup vers $[\not t,\kappa]$, Eve applique une transition $pop(r)$ pour un état $r \in R_\theta^\star$, avec $\star = -$ si $\kappa > 0$ et $\star = +$ sinon. Le lemme 8.1 montrera qu'un tel état r existe.
  Si le coup donné par φ est d'aller vers un sommet $[(q,\alpha,\overrightarrow{R},\min(\theta,\rho(q))),\kappa]$, Eve applique la transition $skip(q)$.
  Si le coup donné par φ est d'aller vers un sommet $[(p,\alpha,\overrightarrow{R},\theta,q,\beta),\kappa]$, alors Eve applique la transition $push(q,\beta)$.
- **Mise à jour de Π :** Supposons que le coup, joué par Eve ou Adam, a été d'aller de $(p,\alpha\sigma)$ vers une configuration (r,σ). La mise à jour de Π est illustrée dans la figure 8.3 et est expliquée ci-dessous. Eve dépile dans Π jusqu'à trouver une configuration de la forme $[(p',\alpha',\overrightarrow{R'},\theta',p'',\alpha,\overrightarrow{R''}),\kappa']$ qui ne soit pas précédée d'une couleur dans $\{0,\dots,d\}$. Cette configuration appartient donc à la marche simulant l'empilement de α. On appelle h l'augmentation maximale de la pile lorsque α est dans la pile. Enfin, on empile θ dans Π suivi de $[(r,\alpha',\overrightarrow{R'},\min(\theta',\theta,\rho(r))),\kappa']$ si $h < \kappa' - 1$ et de $[(r,\alpha',\overrightarrow{R'^+},\min(\theta',\theta,\rho(r))),\kappa'+1]$ sinon.
  Si le coup, joué par Eve ou Adam, a consisté à aller de $(p,\alpha\sigma)$ vers une configuration $(q,\alpha\sigma)$, Eve met à jour Π en empilant par $[(q,\alpha,\overrightarrow{R},\min(\theta,\rho(q))),\kappa]$.
  Si le coup, joué par Eve ou Adam, a consisté à aller de $(p,\alpha\sigma)$ vers une configuration (q,bau), on pose $[(p,\alpha,\overrightarrow{R},\theta,q,\beta,\overrightarrow{S}),\kappa] = \varphi(StCont(\Pi)\cdot[(p,\alpha,\overrightarrow{R},\theta,q,\beta),\kappa])$. Intuitivement, $\overrightarrow{S}$ décrit les états qu'Eve peut assurer si β est un jour dépilé. Eve met à jour Π en empilant successivement $[(p,\alpha,\overrightarrow{R},\theta,q,\beta),\kappa]$, $[(p,\alpha,\overrightarrow{R},\theta,q,\beta,\overrightarrow{S}),\kappa]$, et $[(q,\beta,\overrightarrow{S},\rho(q)),\kappa-1]$.

Concernant la signification du contenu de Π, on a le résultat suivant.

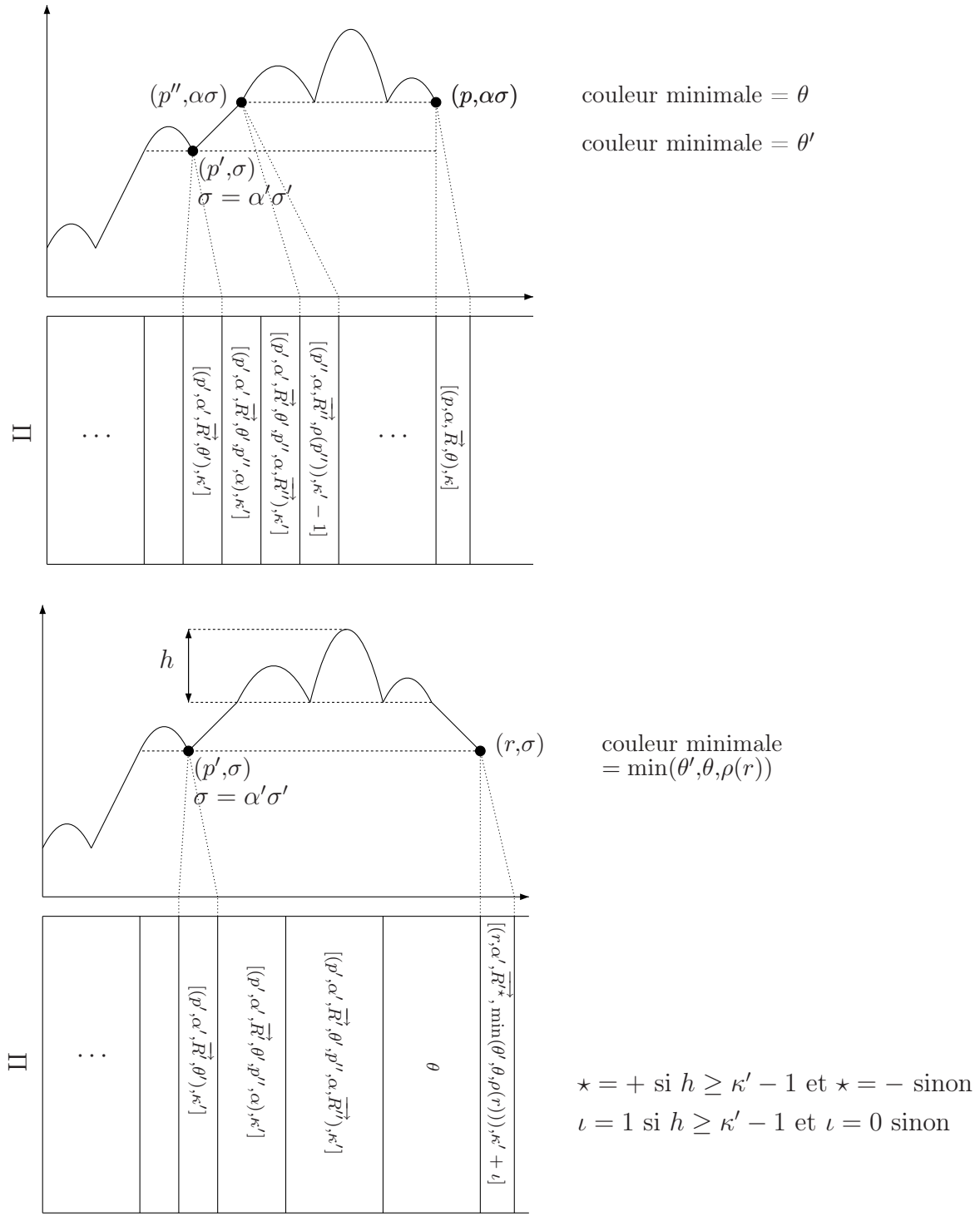


FIG. 8.3 – *Mise à jour de la pile de stratégie* Π

Lemme 8.1 *Soit Λ une partie partielle dans $\mathbb{G}(T)$, où Eve respecte Φ, commençant en $(p_{in},\alpha_{in}\bot)$ et se terminant dans une configuration $(p,\alpha\sigma)$. On a alors les faits suivants :*

1. *top*$(\Pi) = [(p,\alpha,\overrightarrow{R},\theta),\kappa]$ *avec* $R \in \mathcal{P}(Q)^{2d+2}$, $0 \leq \theta \leq d$ *et* $\kappa \geq 0$
2. *StCont*(Π) *est une partie partielle dans* $\widetilde{\mathbb{G}}$ *commençant en* $[(p_{in},\alpha_{in},(T,\ldots,T),\rho(p_{in})),0]$, *se terminant en* $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ *et dans laquelle Eve respecte* φ.
3. θ *est la plus petite couleur visitée depuis que* α *a été empilé.*
4. *si* Λ *est étendue par une transition dépilant le sommet de pile* α, *la configuration atteinte,* (r,σ), *est telle que* $r \in R^{\star}_{\theta}$, *où* $\star = +$ *si la pile avait augmenté d'au moins* $\kappa' - 1$ *symboles depuis que* α *s'y trouvait, et* $\star = -$ *sinon, où* $\kappa' - 1$ *désigne la dernière composante de top*(Π) *suite à l'empilement de* α.

Preuve. La preuve est faite par récurrence sur Λ. On commence par montrer que le dernier point est une conséquence du deuxième et du troisième. Pour une meilleure compréhension, on pourra se référer à la figure 8.3. Supposons donc que la transition après $(p,\alpha\sigma)$ soit $pop(r) \in \Delta(p,\alpha)$. On commence par le cas où $\kappa = 0$. Le second point implique que la configuration $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ est gagnante pour Eve dans $\widetilde{\mathbb{G}}$. Si $p \in Q_{\mathbf{E}}$, par définition de Φ, il y a un arc de cette dernière vers $[\not\!\!t,\kappa]$, ce qui signifie que $r \in R^{+}_{\theta}$ et nous permet de conclure, car on a toujours $\kappa \geq \kappa'$. Si $p \in Q_{\mathbf{A}}$, on ne peut avoir d'arc de la configuration gagnante pour Eve $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ vers $[ff,\kappa]$. On conclut donc de la même façon.

On suppose maintenant que $\kappa > 0$. On appelle h la hauteur maximale dont la pile avait augmenté pendant que α était dedans. On note $[(p'',\alpha,\overrightarrow{R''},\rho(p'')),\kappa' - 1]$ la valeur de *top*(Π) juste après que α ait été empilé. En considérant la manière dont Π est mise à jour, on en conclut qu'alors $\overrightarrow{R} = \overrightarrow{R''}$ si $h < \kappa' - 1$ et $\overrightarrow{R} = \overrightarrow{R''^{+}}$ sinon. Dès lors, il suffit de montrer que $r \in R^{-}_{\theta}$. Le second point implique que la configuration $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ est gagnante pour Eve dans $\widetilde{\mathbb{G}}$. Si $p \in Q_{\mathbf{E}}$, par définition de Φ, il y a un arc de cette dernière vers $[\not\!\!t,\kappa]$, ce qui signifie que $r \in R^{-}_{\theta}$ (car $\kappa > 0$) et permet de conclure. Si $p \in Q_{\mathbf{A}}$, on ne peut avoir d'arc de la configuration gagnante pour Eve $[(p,\alpha,\overrightarrow{R},\theta),\kappa]$ vers $[ff,\kappa]$. On conclut alors de la même façon.

Supposons maintenant que le résultat soit établi pour une partie Λ, et considérons une extension Λ' de Λ. On a deux cas, selon la nature de la transition étendant Λ en Λ' :

- Λ' est obtenu en jouant une règle de type *skip* ou de type *push*. Dans ce cas, le résultat est trivial.
- Λ' est obtenu en jouant une règle de type *pop*. Appelons $(p,\alpha\sigma)$ la dernière configuration de Λ, et soit $\overrightarrow{R}$ la dernière composante vectorielle de *top*(Π) dans $(p,\alpha\sigma)$. Par hypothèse de récurrence, on a $\Lambda' = \Lambda \cdot (r,\sigma)$ avec $r \in R^{\star}_{\theta}$. De la manière dont Π est mise à jour, et grâce au dernier point, on déduit facilement que la nouvelle pile de stratégie Π a la propriété voulue (on se référera pour cela à la figure 8.3).

□

On a en fait un résultat plus précis.

Lemme 8.2 *Soit Λ une partie partielle dans $\mathbb{G}(T)$, où Eve respecte Φ, commençant en $(p_{in},\alpha_{in}\bot)$. Soit $\lambda = StCont(\Pi)$, où Π est le contenu de la pile de stratégie dans la dernière position de Λ. Soit $(\Lambda_i)_{i=0,\ldots,h}$ et $(\lambda_i)_{i=0,\ldots,k}$ les M/B factorisations respectives de Λ et λ. On a alors :*

- *$h = k$.*
- *pour tout $i = 0,\ldots,h$, Λ_i et λ_i ont même nature (marche ou bosse) et même couleur.*
- *pour tout $i = 0,\ldots,h$, pour tout $\kappa \geq 0$, si λ_i est une bosse de hauteur supérieure à κ il en va de même pour Λ_i.*
- *pour tout $i = 0,\ldots,h$, pour tout $\kappa \geq 0$, si λ_i est une bosse de hauteur inférieure à κ il en va de même pour Λ_i.*

Les lemmes 8.1 et 8.2 s'appliquent aux parties partielles. Une version pour les parties infinies permettrait de conclure. Pour cela, on définit une version infinie de $\lambda = StCont(\Pi)$ en considérant la limite au sens des limites de pile. On voit facilement que cette dernière est infinie et qu'elle correspond à une partie remportée par Eve dans $\widetilde{\mathbb{G}}$. De plus, on peut appliquer les résultats du lemme 8.2.

Considérons une partie Λ dans $\mathbb{G}(T)$ commençant en $(p_{in},\alpha_{in}\bot)$, où Eve respecte Φ, et appelons λ sa partie associée dans $\widetilde{\mathbb{G}}$. D'après ce qui précède, λ est remportée par Eve. D'après le lemme 8.2, λ vérifiant la condition de parité, il en est de même pour Λ. De plus, la pile explose dans Λ. En effet, soit il y a une infinité de marches dans λ et l'on conclut directement, soit λ ne contient qu'un nombre fini de marches. Dès lors, on répète infiniment souvent la couleur B, c'est-à-dire que pour tout κ, on fait un jour une bosse de hauteur supérieure à κ. Dès lors, on conclut que Λ visite infiniment souvent un niveau et fait sur ce dernier des bosses arbitrairement hautes. D'où le résultat.

8.2.4 Autres combinaisons booléennes

On s'intéresse maintenant aux autres combinaisons booléennes prises comme condition de gain. On appelle $\mathbb{G}$ le jeu sur $\mathcal{G}$ muni d'une condition de gain Ω qui est une combinaison booléenne d'une condition de parité et d'une condition sur la hauteur de pile. On appelle $\widetilde{\mathbb{G}}$ le jeu sur $\widetilde{\mathcal{G}}$ équipé de la condition de gain $\widetilde{\Omega}$, où $\widetilde{\Omega}$ est donnée dans la table 8.1 en fonction de Ω.

On a alors le résultat suivant qui généralise le théorème 8.6

Théorème 8.7 *Soit $p_{in} \in Q$, soit $\alpha_{in} \in \Gamma$ et soit $T \subseteq Q$. On a les équivalences suivantes :*

- *Eve possède une stratégie gagnante dans $\mathbb{G}(T)$ depuis $(p_{in},\alpha_{in}\bot)$ si et seulement si elle possède une stratégie gagnante dans $\widetilde{\mathbb{G}}$ depuis $[(p_{in},\alpha_{in},(T,\ldots,T),\rho(p_{in})),0]$.*
- *Eve possède une stratégie gagnante dans $\mathbb{G}$ depuis $(p_{in},\bot)$ si et seulement si elle possède une stratégie gagnante dans $\widetilde{\mathbb{G}}$ depuis $[(p_{in},\bot,(\emptyset,\ldots,\emptyset),\rho(p_{in})),0]$.*

La preuve du théorème 8.7 est une simple adaptation de celle du théorème 8.7.

Nature de Ω	Nature de $\widetilde{\Omega}$
Parité et explosion	Parité et infiniment dans $\{m,B\}$
Parité ou explosion	Parité ou infiniment dans $\{m,B\}$
Parité et bornage	Parité et finiment dans $\{m,B\}$
Parité ou bornage	Parité ou finiment dans $\{m,B\}$
Parité et explosion stricte	Parité et infiniment dans $\{m\}$
Parité ou explosion stricte	Parité ou infiniment dans $\{m\}$
Parité et répétition	Parité et finiment dans $\{m\}$
Parité ou répétition	Parité ou finiment dans $\{m\}$

TAB. 8.1 – *Nature de* $\widetilde{\Omega}$

8.2.5 Résolution

D'après le théorème 8.7, si l'on sait résoudre les différents jeux sur $\widetilde{\mathcal{G}}$, on sait résoudre n'importe quel jeu sur $\mathcal{G}$ muni d'un condition qui est une combinaison booléenne d'une condition de parité et d'une condition sur la hauteur de pile.

Les conditions sur $\widetilde{\mathcal{G}}$ sont toujours une disjonction ou une conjonction d'une condition de parité et d'une condition de Büchi ou de co-Büchi. Il n'est pas très difficile de voir que l'on peut à chaque fois modifier $\widetilde{\mathcal{G}}$ de façon à obtenir un graphe de jeu *équivalent* $\widetilde{\mathcal{G}}'$ qui soit équipé d'une condition de parité.

On se contente de donner l'idée pour les deux premiers cas. Pour ce qui est des deux autres cas, il suffit de considérer le problème du point de vue d'Adam : Les conditions de gain sont alors les conditions duales et l'on retrouve alors les deux premiers cas.

- si $\widetilde{\Omega}$ est une conjonction d'une condition de parité et d'une condition de Büchi, on modifie le processus à compteur en rajoutant une composante qui se rappelle de la plus petite couleur visitée. A chaque état final est associée la couleur précédemment stockée, couleur qui est alors réinitialisée. Tous les autres états sont colorés par une nouvelle couleur choisie impaire et maximale. Une partie ne visitant pas infiniment souvent des état finaux ne visitera infiniment que la nouvelle couleur et sera donc perdante. Sinon, la plus petite couleur infiniment souvent visitée est la même qu'avant.
- si $\widetilde{\Omega}$ est une disjonction d'une condition de parité et d'une condition de Büchi, on ajoute une couleur, que l'on choisit paire et minimale. On l'associe à tout état final. Ainsi, une partie visitant infiniment souvent des état finaux aura cette nouvelle couleur comme plus petite couleur et sera donc gagnante. Sinon, la plus petite couleur infiniment souvent visitée est la même qu'auparavant.

Dès lors en appliquant l'algorithme PSPACE décrit dans le chapitre 7 pour les jeux de parité sur des graphes de processus à compteur on obtient un algorithme global en EXPSPACE. On a donc prouvé le résultat qui suit.

Théorème 8.8 *Soit un jeu* $\mathbb{G}$ *sur un graphe de processus à pile muni d'une condition de gain qui est combinaison booléenne d'une condition de parité et d'une condi-*

tion sur la hauteur de pile. Décider le gagnant dans $\mathbb{G}$ *se réduit à décider le gagnant dans un jeu de parité sur un graphe de processus à pile exponentiellement plus grand.*

En particulier, on peut décider le gagnant dans $\mathbb{G}$ *en* EXPSPACE.

8.2.6 Une borne supérieure en DEXPTIME

On présente ici les résultats obtenus avec A.J. Bouquet et I. Walukiewicz dans [13]. Nous nous sommes intéressés à la résolution des jeux sur des graphes de processus à pile munis d'une condition qui est une combinaison booléenne d'une condition de Büchi et d'une condition d'explosion. Nous avons montré que décider le gagnant dans un tel jeu est un problème DEXPTIME-complet. Les preuves pour les conditions qui sont une disjonction d'une condition de Büchi ou de co-Büchi avec une condition d'explosion ne sont pas très complexes et peuvent être étendues au cas d'une condition de la forme parité ou explosion, même si cela entraîne un alourdissement substantiel des notations. Les choses se compliquent fortement pour le cas des conjonctions d'une condition de Büchi ou de co-Büchi avec la condition d'explosion. Cette difficulté vient principalement du fait que les stratégies gagnantes ne sont plus positionnelles et que les deux conditions de gains sont dès lors intimement mêlées.

Bien que les résultats obtenus dans [13] soient meilleurs que ceux donnés dans le théorème 8.8, nous avons privilégié ces derniers pour plusieurs raisons. Tout d'abord, ils sont plus simples à présenter : la preuve est une généralisation de celles données pour la condition de parité et pour la condition d'explosion. Ensuite, ils fournissent une conséquence intéressante du théorème 7.3 pour les jeux de parité sur les graphes de processus à compteur. Ils sont aussi plus complets car ils traitent des conditions de type parité et explosion stricte. Enfin, la preuve est la même pour toutes les conditions de gain, ce qui n'est malheureusement pas le cas dans [13]. Nous avons donc fait le choix de la concision et de la généralité au détriment d'une complexité optimale.

Remarque 8.4 Bien que la présentation des résultats obtenus dans [13] soit complexe, nous espérons aboutir prochainement à une preuve simple et homogène. Par ailleurs, une extension de nos techniques au cas des conditions de la forme parité et explosion nous semble envisageable.

Nous présentons brièvement les constructions utilisées dans [13] pour obtenir une borne en DEXPTIME. Pour une version détaillée, on consultera [13, 12].

Dans tout ce qui suit, on considère un processus à pile $\mathcal{P} = \langle Q,\Gamma,\bot,\Delta\rangle$. On suppose qu'il n'y a pas de transition de type *skip* dans Δ, ce qui n'est pas une restriction ici. On appelle $G = (V,E)$ le graphe qu'il engendre, on se donne une partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ de Q, et l'on considère le graphe de jeu $\mathcal{G} = (G,V_{\mathbf{E}},V_{\mathbf{A}})$ associé à G et à cette partition de Q. Enfin, on considère une fonction de coloriage ρ de Q dans un ensemble fini de couleurs. La fonction ρ permet donc de définir une condition de Büchi ou une condition de co-Büchi selon que les couleurs sont $\{0,1\}$ ou $\{1,2\}$.

Büchi ou explosion

On considère une condition de type Büchi ou explosion sur $\mathcal{G}$, et l'on appelle $\mathbb{G}$ le jeu associé. Enfin, on considère un sous-ensemble T de Q. On souhaite décider si

une configuration (p_{in},α_{in}) est gagnante pour Eve dans $\mathbb{G}$. L'idée est de reprendre les réductions données pour la condition de parité (la condition de Büchi est une condition de parité sur les couleurs 0 et 1) et pour la condition d'explosion stricte. En mélangeant les deux constructions, on définit un graphe de jeu $\widetilde{\mathcal{G}}$ sur un graphe fini qui est illustré dans la figure 8.4.

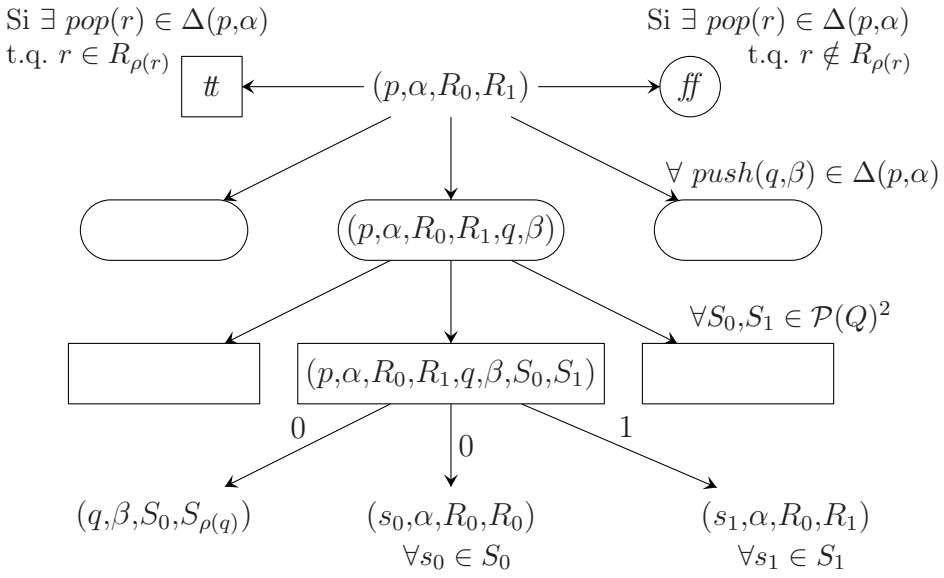


FIG. 8.4 – *Structure de $\widetilde{\mathcal{G}}$ pour la condition Büchi ou explosion.*

Le sommets de $\widetilde{\mathcal{G}}$ sont comme sur la figure, pour tout $p,q \in Q$, tout $\alpha,\beta \in \Gamma$, et tout $R_0,R_1,S_0,S_1 \in \mathcal{P}(Q)$. Depuis (p,α,R_0,R_1), il y a un arc vers *tt* si et seulement s'il existe une règle $pop(r) \in \Delta(p,\alpha)$ avec $r \in R_{\rho(r)}$, et un arc vers *ff* si et seulement s'il existe une règle $pop(r) \in \Delta(p,\alpha)$ avec $r \notin R_{\rho(r)}$. Depuis (p,α,R_0,R_1), il y a un arc vers $(p,\alpha,R_0,R_1,q,\beta)$ pour toute règle $push(q,\beta) \in \Delta(p,\alpha)$. Depuis $(p,\alpha,R_0,R_1,q,\beta)$, il y a un arc vers $(p,\alpha,R_0,R_1,q,\beta,S_0,S_1)$ pour tout sous-ensemble $S_0,S_1 \in \mathcal{P}(Q)$. Enfin, depuis $(p,\alpha,R_0,R_1,q,\beta,S_0,S_1)$, il y a un arc étiqueté par 0 vers $(q,\beta,S_0,S_{\rho(q)})$, un arc étiqueté par 0 vers (s_0,α,R_0,R_0) pour tout $s_0 \in S_0$, et un arc étiqueté par 1 vers (s_1,α,R_0,R_1) pour tout $s_1 \in S_1$.

Les sommets appartenant à Eve sont ceux de la forme (p,α,R_0,R_1) avec $p \in Q_{\mathbf{E}}$, ceux de la forme $(p,\alpha,R_0,R_1,q,\beta)$, et le sommet *ff*.

On appelle $\widetilde{\mathbb{G}}$ le jeu de Büchi sur le graphe de jeu $\widetilde{\mathcal{G}}$ et l'on a alors le résultat qui suit.

Théorème 8.9 *[13] Une configuration $(q_{in},\alpha_{in}\bot)$ est gagnante pour Eve dans $\mathbb{G}(T)$ si et seulement si la configuration (q_{in},α_{in},T,T) est gagnante pour Eve dans $\widetilde{\mathbb{G}}$.*

co-Büchi ou explosion

On considère une condition de type co-Büchi et explosion sur $\mathcal{G}$, et l'on appelle $\mathbb{G}$ le jeu associé. Enfin, on considère un sous-ensemble T de Q. On souhaite décider si une configuration $(p_{in},\alpha_{in}\bot)$ est gagnante pour Eve dans $\mathbb{G}$. A nouveau, l'idée

est de reprendre les réductions utilisées pour la condition de parité (la condition de co-Büchi est une condition de parité sur les couleurs 1 et 2) et pour la condition d'explosion stricte. En mélangeant les deux constructions, on définit un graphe de jeu $\widetilde{\mathcal{G}}$ sur un graphe fini qui est illustré dans la figure 8.5.

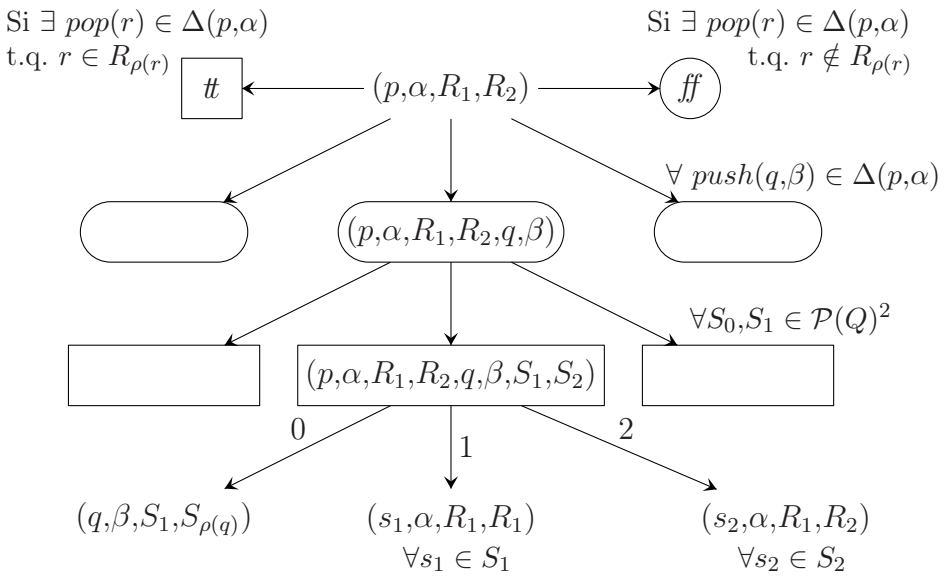


FIG. 8.5 – *Structure de $\widetilde{\mathcal{G}}$ pour la condition co-Büchi ou explosion.*

Le sommets de $\widetilde{\mathcal{G}}$ sont comme sur la figure, pour tout $p,q \in Q$, tout $\alpha,\beta \in \Gamma$, et tout $R_1,R_2,S_1,S_2 \in \mathcal{P}(Q)$. Depuis (p,α,R_1,R_2), il y a un arc vers *tt* si et seulement s'il existe une règle $pop(r) \in \Delta(p,\alpha)$ avec $r \in R_{\rho(r)}$, et un arc vers *ff* si et seulement s'il existe une règle $pop(r) \in \Delta(p,\alpha)$ avec $r \notin R_{\rho(r)}$. Depuis (p,α,R_1,R_2), il y a un arc vers $(p,\alpha,R_1,R_2,q,\beta)$ pour toute règle $push(q,\beta) \in \Delta(p,\alpha)$. Depuis $(p,\alpha,R_1,R_2,q,\beta)$, il y a un arc vers $(p,\alpha,R_1,R_2,q,\beta,S_1,S_2)$ pour tout sous-ensemble $S_1,S_2 \in \mathcal{P}(Q)$. Enfin, depuis $(p,\alpha,R_1,R_2,q,\beta,S_1,S_2)$, il y a un arc étiqueté par 0 vers $(q,\beta,S_1,S_{\rho(q)})$, un arc étiqueté par 1 vers (s_1,α,R_1,R_1) pour tout $s_1 \in S_1$, et un arc étiqueté par 2 vers (s_2,α,R_1,R_2) pour tout $s_2 \in S_2$.

Les sommets appartenant à Eve sont ceux de la forme (p,α,R_1,R_2) avec $p \in Q_{\mathbf{E}}$, ceux de la forme $(p,\alpha,R_1,R_2,q,\beta)$, et le sommet *ff*.

On appelle $\widetilde{\mathbb{G}}$ le jeu de co-Büchi sur le graphe de jeu $\widetilde{\mathcal{G}}$ et l'on a alors le résultat qui suit.

Théorème 8.10 *[13] Une configuration $(q_{in},\alpha_{in}\bot)$ est gagnante pour Eve dans $\mathbb{G}(T)$ si et seulement si la configuration (q_{in},α_{in},T,T) est gagnante pour Eve dans $\widetilde{\mathbb{G}}$.*

co-Büchi et explosion

On considère une condition de type co-Büchi et explosion sur $\mathcal{G}$, et l'on appelle $\mathbb{G}$ le jeu associé. Enfin, on considère un sous-ensemble T de Q. On souhaite décider si une configuration $(p_{in},\alpha_{in}\bot)$ est gagnante pour Eve dans $\mathbb{G}$. A nouveau, l'idée

est de reprendre les réductions utilisées pour la condition de parité (la condition de co-Büchi est une condition de parité sur les couleurs 1 et 2) et pour la condition d'explosion stricte. En mélangeant les deux constructions, on définit un graphe de jeu $\widetilde{\mathcal{G}}$ sur un graphe fini qui est illustré dans la figure 8.6.

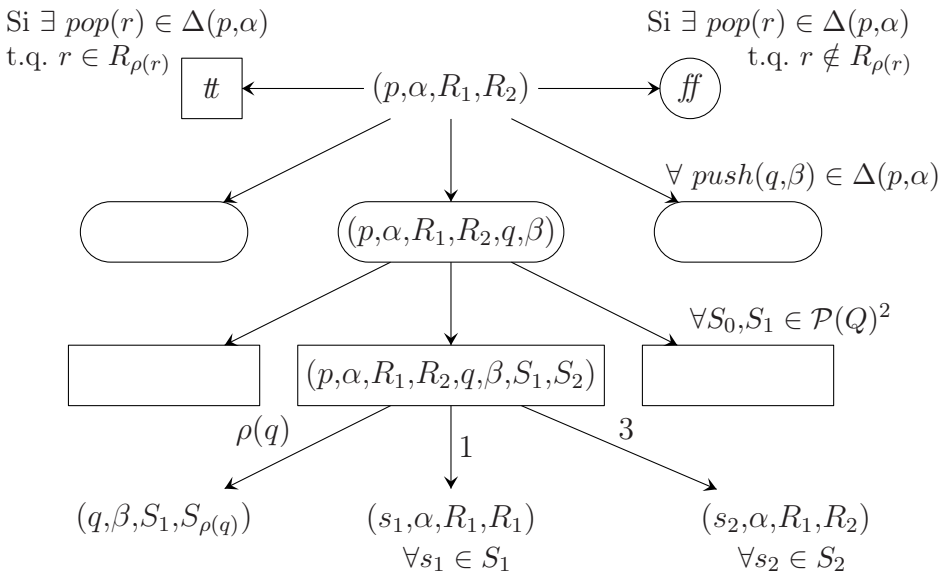


FIG. 8.6 – *Structure de $\widetilde{\mathcal{G}}$ pour la condition co-Büchi et explosion.*

Le sommets de $\widetilde{\mathcal{G}}$ sont comme sur la figure, pour tout $p,q \in Q$, tout $\alpha,\beta \in \Gamma$, et tout $R_1,R_2,S_1,S_2 \in \mathcal{P}(Q)$. Depuis (p,α,R_1,R_2), il y a un arc vers *tt* si et seulement s'il existe une règle $pop(r) \in \Delta(p,\alpha)$ avec $r \in R_{\rho(r)}$, et un arc vers *ff* si et seulement s'il existe une règle $pop(r) \in \Delta(p,\alpha)$ avec $r \notin R_{\rho(r)}$. Depuis (p,α,R_1,R_2), il y a un arc vers $(p,\alpha,R_1,R_2,q,\beta)$ pour toute règle $push(q,\beta) \in \Delta(p,\alpha)$. Depuis $(p,\alpha,R_1,R_2,q,\beta)$, il y a un arc vers $(p,\alpha,R_1,R_2,q,\beta,S_1,S_2)$ pour tout sous-ensemble $S_1,S_2 \in \mathcal{P}(Q)$. Enfin, depuis $(p,\alpha,R_1,R_2,q,\beta,S_1,S_2)$, il y a un arc étiqueté par $\rho(q)$ vers $(q,\beta,S_1,S_{\rho(q)})$, un arc étiqueté par 1 vers (s_1,α,R_1,R_1) pour tout $s_1 \in S_1$, et un arc étiqueté par 3 vers (s_2,α,R_1,R_2) pour tout $s_2 \in S_2$.

Les sommets appartenant à Eve sont ceux de la forme (p,α,R_1,R_2) avec $p \in Q_{\mathbf{E}}$, ceux de la forme $(p,\alpha,R_1,R_2,q,\beta)$, et le sommet *ff*.

On appelle $\widetilde{\mathbb{G}}$ le jeu de parité (à trois couleurs) sur le graphe de jeu $\widetilde{\mathcal{G}}$ et l'on a alors le résultat qui suit.

Théorème 8.11 *[13] Une configuration $(q_{in},\alpha_{in}\bot)$ est gagnante pour Eve dans $\mathbb{G}(T)$ si et seulement si la configuration (q_{in},α_{in},T,T) est gagnante pour Eve dans $\widetilde{\mathbb{G}}$.*

Büchi et explosion

On considère une condition de type Büchi et explosion sur $\mathcal{G}$, et l'on appelle $\mathbb{G}$ le jeu associé. Enfin, on considère un sous-ensemble T de Q. On souhaite décider si une configuration $(p_{in},\alpha_{in}\bot)$ est gagnante pour Eve dans $\mathbb{G}$.

Nous avons vu dans la proposition 8.3 que l'on peut avoir besoin d'une mémoire infinie pour gagner dans un jeu muni d'une condition de la forme Büchi et explosion. La résolution de ce cas est donc différente des autres. Il est clair que si Eve peut gagner pour la condition de Büchi et explosion, elle peut également gagner pour la condition de Büchi d'une part et pour la condition d'explosion d'autre part. Plus précisément, elle peut, tout en gagnant pour l'une de ces conditions, rester dans l'ensemble des positions gagnantes pour la seconde. L'idée alors naturelle est la suivante : Eve doit pouvoir gagner pour une condition où c'est Adam qui décide si Eve doit visiter infiniment souvent des états finaux ou exploser la pile. De plus, ce choix d'Adam peut être modifié aussi souvent qu'il le veut, à condition qu'il ne change qu'un nombre fini d'avis. En effet, le fait qu'Adam puisse changer d'avis force Eve à rester dans l'ensemble des positions gagnantes pour la condition qui n'est pas choisie. Le fait qu'Adam ne change que finiment souvent la condition à satisfaire permet d'éviter que celui-ci ne laisse pas le temps à Eve de faire ses preuves.

On définit à partir de $\mathcal{G}$ un nouveau graphe de jeu $\mathcal{G}^*$ illustré dans la figure 8.7.

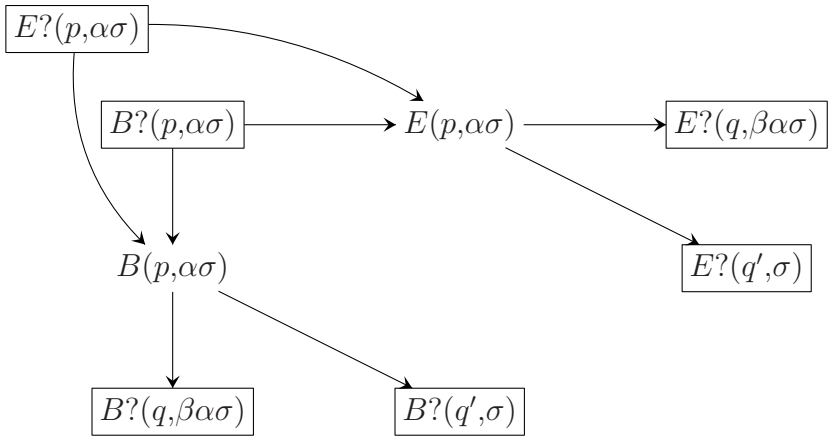


FIG. 8.7 – *Structure de* $\mathcal{G}^*$.

On qualifie de modes les éléments de $\{B,E\}$: B pour Büchi et E pour Explosion. Les sommets de $\mathcal{G}^*$ sont obtenus de la façon suivante : pour tout mode K et toute configuration (p,σ) dans le graphe de jeu $\mathcal{G}$, il y a des sommets $K(p,\sigma)$ et $K?(p,\sigma)$ dans $\mathcal{G}^*$. Pour ce qui est des arcs dans $\mathcal{G}^*$, on a, pour tout mode K, un arc de $K?(p,\sigma)$ vers $K(p,\sigma)$ et $\overline{K}(p,\sigma)$, où $\overline{K}$ désigne B si $K = E$ et E sinon. Enfin, pour tout arc dans $\mathcal{G}$ d'une configuration (p,σ) vers une configuration (q,ν), on a, pour tout mode K, un arc de $K(p,\sigma)$ vers $K?(q,\nu)$.

Les positions avec un point d'interrogation appartiennent à Adam. Une position $K(p,\sigma)$ appartient à Eve si et seulement si $p \in Q_{\mathbf{E}}$.

Enfin, on considère la condition de gain Ω^* formée des parties infinies $\Lambda = K_1?(p_1,\sigma_1)K_2(p_1,\sigma_1)K_2?(p_2,\sigma_2)K_3(p_2,\sigma_2)K_3?(p_3,\sigma_3)\cdots$ telles que :

- il existe un indice $i \geq 1$ tel que pour tout $j \geq i$, $K_j = E$, et la pile explose dans $(p_1,\sigma_1)(p_2,\sigma_2)(p_3,\sigma_3)\cdots$.
- il existe un indice $i \geq 1$ tel que pour tout $j \geq i$, $K_j = B$, et la condition de Büchi est satisfaite dans $(p_1,\sigma_1)(p_2,\sigma_2)(p_3,\sigma_3)\cdots$.

– il existe une infinité d'indices $i \geq 1$ tels que $K_i \neq K_{i+1}$.

On considère le jeu $\mathbb{G}^*$ sur $\mathcal{G}^*$ dans lequel une partie finie est perdue par le joueur bloqué, et une partie infinie est remportée par Eve si et seulement si elle appartient à Ω^*.

On a alors le résultat suivant.

Proposition 8.7 *[13] Eve possède une stratégie gagnante depuis une configuration (p,σ) dans $\mathbb{G}$ si et seulement si elle possède une stratégie gagnante depuis les configurations $B?(p,\sigma)$ et $E?(p,\sigma)$ dans $\mathbb{G}^*$.*

On reprend à présent les réductions utilisées pour la condition de parité et pour la condition d'explosion stricte, et l'on les adapte à $\mathbb{G}^*$. En mélangeant les deux constructions, on définit un graphe de jeu $\widetilde{\mathcal{G}}^*$ sur un graphe fini qui est illustré dans la figure 8.8.

On appelle $Res = Q \times \Gamma \times \mathcal{P}(Q)^2$. Les sommets de $\mathcal{G}^*$ sont : $K(\eta)$, $K?(\eta)$, $K(\eta,q,\beta)$, $K(\eta,\eta')$, tt et ff, pour tout $K \in \{B,E\}$, $\eta,\eta' \in Res$, $q \in Q$ et $\beta \in \Gamma$. Parmi ceux-ci, les sommets appartenant à Eve soit ceux de la forme $K(\eta)$ pour tout $\eta \in Q_{\mathbf{E}} \times \Gamma \times \mathcal{P}(Q)^2$, $K(\eta,q,\beta)$ pour tout $\eta \in Res$, $q \in Q$ et $\beta \in \Gamma$, et ff.

Depuis $K(p,\alpha,R_0,R_1)$, pour $K \in \{B,E\}$, il y a un arc vers tt si et seulement s'il existe une règle $pop(r) \in \Delta(p,\alpha)$ avec $r \in R_{\rho(r)}$, et un arc vers ff si et seulement s'il existe une règle $pop(r) \in \Delta(p,\alpha)$ avec $r \notin R_{\rho(r)}$.

Depuis $K(p,\alpha,R_0,R_1)$, pour $K \in \{B,E\}$, il y a un arc vers $K(p,\alpha,R_0,R_1,q,\beta)$ pour toute règle $push(q,\beta) \in \Delta(p,\alpha)$. Depuis $K(p,\alpha,R_0,R_1,q,\beta)$, il y a un arc vers $K(p,\alpha,R_0,R_1,q,\beta,S_0,S_1)$ pour tout sous-ensemble $S_0,S_1 \in \mathcal{P}(Q)$. Enfin, depuis $K(p,\alpha,R_0,R_1,q,\beta,S_0,S_1)$ il y a un arc, étiqueté par 0 si $K = E$ et par $\rho(q)$ si $K = B$, vers $K?(q,\beta,S_0,S_{\rho(q)})$, un arc étiqueté par 0 vers $K?(s_0,\alpha,R_0,R_0)$ pour tout $s_0 \in S_0$, et un arc étiqueté par 1 vers $K?(s_1,\alpha,R_0,R_1)$ pour tout $s_1 \in S_1$.

On appelle $\widetilde{\mathbb{G}}^*$ le jeu de Büchi sur le graphe de jeu $\widetilde{\mathcal{G}}^*$ et l'on a alors le résultat suivant :

Théorème 8.12 *[13] Une configuration $(q_{in},\alpha_{in}\bot)$ est gagnante pour Eve dans $\mathbb{G}^*(T)$ si et seulement si la configuration (q_{in},α_{in},T,T) est gagnante pour Eve dans $\widetilde{\mathbb{G}}^*$.*

Complexité

Dans chacun des quatre cas ci-dessus, nous avons donc proposé une réduction vers un jeu de parité (avec au plus trois couleurs) sur un graphe fini de taille exponentielle, qui permet de calculer les ensembles de retours dans le jeu $\mathbb{G}$ de départ. Les jeux sur les graphes finis peuvent ensuite être résolus en temps polynomial dans la taille du graphe de jeu, et l'on obtient donc le résultat suivant (la complétude ne pose pas de problème).

Théorème 8.13 *[13] Décider le gagnant dans un jeu sur un graphe de processus à pile muni d'une condition de gain de la forme Büchi/co-Büchi et/ou explosion est un problème* Dexptime*-complet.*

Ces résultats ont été étendus et établis indépendamment par H. Gimbert dans [38]. Les constructions reposent sur la notion d'automate d'arbre et généralisent les techniques de M. Vardi et O. Kupferman [50].

Théorème 8.14 *[38] Décider le gagnant dans un jeu sur un graphe de processus à pile muni d'une condition de gain de la forme parité et/ou explosion est un problème* DEXPTIME*-complet.*

Stratégies

Dans [13], nous avons également montré le résultat suivant concernant les stratégies gagnantes.

Théorème 8.15 *[13] Dans un jeu sur un graphe de processus à pile muni d'une condition de gain de la forme Büchi/co-Büchi et/ou explosion, les deux joueurs possèdent des stratégies à pile depuis toute position gagnante.*

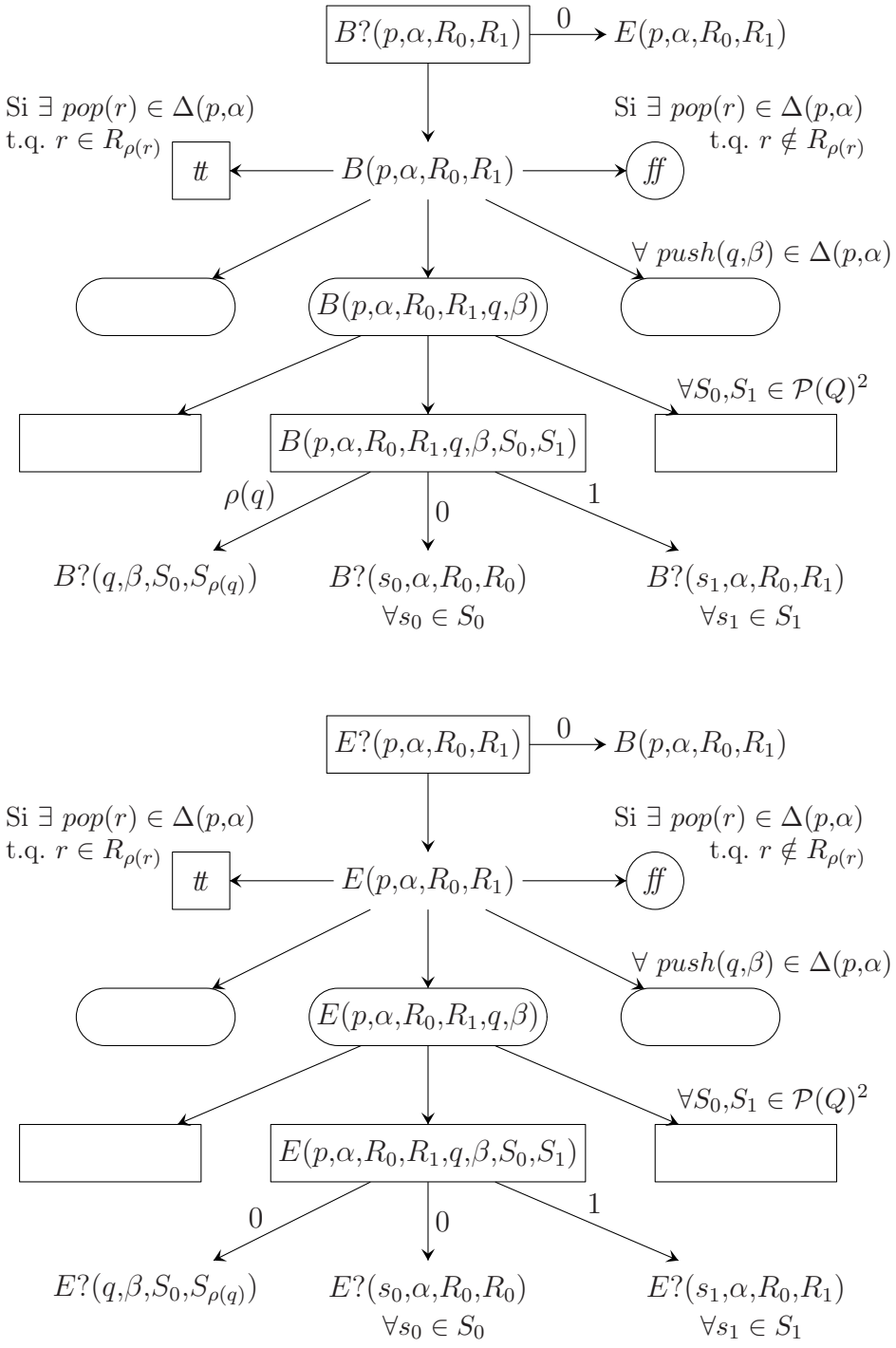


FIG. 8.8 – *Structure de* $\widetilde{\mathcal{G}}^*$.

Conclusion

Ce travail de recherche avait pour but d'étudier différentes conditions de gains pour des jeux sur des graphes de processus à pile et leurs variantes. En conclusion, récapitulons les résultats obtenus et ouvrons quelques perspectives.

Récapitulatif des résultats

Le chapitre 4 avait pour objet de donner une grande famille de conditions de gain induisant des ensembles de positions gagnantes réguliers. Pour ces conditions nous avons montré comment construire un $\mathcal{P}$-automate alternant reconnaissant l'ensemble des positions gagnantes. Cette construction est effective à condition de pouvoir calculer les ensembles de retours et décider qui gagne depuis les configurations de pile vide. A l'exception des conditions du chapitre 6, toutes les conditions étudiées induisent des ensembles de positions gagnantes régulières. Dès lors, l'étude de ces jeux se réduit au calcul des ensembles de retours et des positions gagnante de pile vide. Enfin, les techniques du chapitre 4 ont permis de calculer des stratégies gagnantes à pile pour de nombreux jeux.

Le chapitre 5 a été consacrée à une présentation nouvelle des résultats d'I. Walukiewicz concernant les jeux de parité sur des processus à pile. Il a alors été facile d'adapter ces méthodes aux jeux d'explosion stricte et aux jeux de parité en escalier. Nous avons par ailleurs montré que les conditions d'explosion et d'explosion stricte coïncident et qu'il en était de même pour les conditions de répétition et de bornage. Toutes les conditions proposées dans ce chapitre ont conduit à des algorithmes Dexptime. Enfin, nous avons montré que cette borne est aussi une borne inférieure.

Le chapitre 6 s'est intéressé à un problème plus théorique, celui de l'existence de conditions de gain décidables de complexité borélienne arbitraire finie. Nous y avons montré que décider le gagnant dans un tel jeu est non-élémentaire et est Elementary-dur.

Le chapitre 7 s'est quant à lui intéressé à des jeux de parité sur des graphes de jeu plus simples. En se penchant sur le cas des graphes de Bpa, nous avons proposé deux définitions possibles de graphes de jeu, puis étudié les jeux de parité sur des graphes de processus à compteur. Pour ces derniers jeux, nous avons utilisé une méthode tout à fait différente des autres, inspirée des travaux de O. Kupferman et M. Vardi. Nous avons alors montré que l'on peut décider le gagnant en Pspace,

ce qui implique que le problème du model-checking du μ-calcul sur un graphe de processus à compteur est PSPACE. Enfin, nous avons montré que ces problèmes sont DP-dur.

Enfin, le dernier chapitre a été consacré à donner une autre application des résultats pour les jeux de parité sur des graphes de processus à compteur. Nous avons tout d'abord obtenu un algorithme en EXPSPACE pour les jeux munis d'une condition de gain qui est une combinaison booléenne d'une condition de parité et d'une condition sur la hauteur de pile. De plus, nous avons donné une borne en DEXPTIME lorsque la condition de gain est une combinaison booléenne d'une condition de Büchi et d'une condition d'explosion. Auparavant, nous avions également donné pour les jeux munis d'une condition de gain ω-VPL des bornes optimales de complexité.

La table 8.2 résume les différents résultats obtenus.

Perspectives

Quelques perspectives ont déjà été introduites dans le chapitre 6. D'une part ces résultats pourraient être étendus à des conditions de gain de complexité borélienne non finie, voire à des conditions de gain non boréliennes. D'autre part, les constructions pourraient être symétrisées. Enfin, la nature exacte des ensembles de positions gagnantes mériterait d'être précisée. Le chapitre 7 a donné une nouvelle borne supérieure pour le problème du model-checking du μ-calcul pour les processus à compteur, mais nous ne savons toujours pas si cette borne est optimale. C'est un problème d'autant plus intéressant qu'il possède des liens très naturels avec la vérification. Enfin, les résultats du chapitre 8 se situent dans un cadre plus général, celui de la recherche de conditions de gain naturelles pour la vérification, pour lesquelles on peut décider le gagnant. Une remarque s'impose d'ailleurs à ce sujet : si l'on est capable de décider le gagnant pour une condition de gain externe sur tout graphe de jeu, on peut alors décider si la condition de gain est vide ou pleine. Par exemple, si l'on considère une condition de gain donnée par un automate à pile non déterministe de Büchi, on ne pourra déterminer le gagnant, puisque l'universalité pour de tels automates est indécidable. A la manière de ce que propose CARET, une approche logique pourrait s'avérer séduisante pour aborder ce problème.

Terminons enfin par l'évocation d'une dernière direction de recherche. Dans [22], D. Caucal a introduit une hiérarchie de graphes infinis finiment représentables qui généralisent les graphes de processus à pile. Il serait intéressant d'étudier les jeux sur de tels graphes comme le fait T. Cachat dans [19] pour les conditions de parité. Il semblerait cependant plus pertinent d'étudier pour ces graphes des conditions de gains plus générales qui exploitent davantage le modèle, un peu à la manière des conditions sur la hauteur de pile pour les jeux sur des graphes de processus à pile.

Graphe de jeu	**Condition de gain**	**Complexité**	**Référence**
Graphe de processus à pile	Parité	Dexptime-complet	[83]
Graphe de processus à pile	Explosion stricte/Répétition	Dexptime-complet	chapitre 5 et [21]
Graphe de processus à pile	Explosion/Bornage	Dexptime-complet	chapitre 5
Graphe de processus à pile	Parité en escalier	Dexptime-complet	chapitre 5 et [56]
Graphe de processus à pile	$\Omega^{int}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$	non élémentaire, Elementary-dur	chapitre 6 et [76, 74]
Graphe fini	$\Omega^{ext}_{\mathcal{A}_1 \triangleright \cdots \triangleright \mathcal{A}_n \triangleright \mathcal{A}_{n+1}}$	non élémentaire, Elementary-dur	chapitre 6 et [76, 74]
Graphe de Bpa	parité locale	comme parité sur graphe fini	chapitre 7
Graphe de Bpa global	parité globale	Dexptime-complet	chapitre 7
Graphe de processus à compteur	parité	Pspace, DP-dur	chapitre 7
Graphe de Vpp	Langage ω-Vpl donné par un StVpa det.	Dexptime-complet	chapitre 8 et [56]
Graphe de Vpp	Langage ω-Vpl donné par un Vpa non det.	2-Dexptime-complet	chapitre 8 et [56]
Graphe de Vpp	Formule Caret	3-Dexptime-complet	chapitre 8 et [56]
Graphe de processus à pile	Formule ltl	3-Dexptime-complet	chapitre 8 et [56]
Graphe de processus à pile	Combinaison Booléenne parité/hauteur de pile	Expspace, Dexptime-dur	chapitre 8
Graphe de processus à pile	Combinaison Booléenne Büchi/explosion	Dexptime-complet	chapitre 8 et [13]
Graphe de processus à pile	Combinaison Booléenne parité/explosion	Dexptime-complet	[38]

Tab. 8.2 – *Récapitulatif des résultats*

Table des figures

Index

Bibliographie

[1] Rajeev Alur, Kousha Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, TACAS 2004, Proceedings*, volume 2988 of *Lecture Notes in Computer Science*, pages 467–481. Springer, 2004.

[2] Rajeev Alur, Salvatore La Torre, and P. Madhusudan. Modular strategies for infinite games on recursive graphs. In *Computer Aided Verification, 15th International Conference, CAV 2003, Proceedings*, volume 2725 of *Lecture Notes in Computer Science*, pages 67–79. Springer, 2003.

[3] Rajeev Alur, Salvatore La Torre, and P. Madhusudan. Modular strategies for recursive game graphs. In *Tools and Algorithms for the Construction and Analysis of Systems, 9th International Conference, TACAS 2003, Proceedings*, volume 2725 of *Lecture Notes in Computer Science*, pages 363–378. Springer, 2003.

[4] Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In *36th Annual ACM Symposium on Theory of Computing, STOC 2004, Proceedings*, pages 202–211. ACM, 2004.

[5] André Arnold, Aymeric Vincent, and Igor Walukiewicz. Games for synthesis of controlers with partial observation. *Theoretical Computer Science*, 303(1):7–34, 2003.

[6] Arnold Arnold and Damian Niwiński. *Rudiments of mu-calculus*, volume 146 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 2001.

[7] Jan. A. Bergstra and Jan Willem Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985.

[8] Julien Bernet, David Janin, and Igor Walukiewicz. Premissive strategies: from parity games to safety games. *R.A.I.R.O. — Informatique Théorique et Applications*, 36:261–275, 2002.

[9] Jean Berstel. *Transduction and context free languages*. Teubner Studienbücher Informatik, 1979.

[10] Luc Boasson and Maurice Nivat. Adherences of languages. *Journal of Computer and System Sciences*, 20(3):285–309, 1980.

[11] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Concurrency Theory, 8th International Conference, CONCUR 1997, Proceedings*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 1997.

[12] Alexis-Julien Bouquet, Olivier Serre, and Igor Walukiewicz. Pushdown games with the unboundedness and regular conditions: full version with complete proofs. `http://www.liafa.jussieu.fr/~serre`.

[13] Alexis-Julien Bouquet, Olivier Serre, and Igor Walukiewicz. Pushdown games with the unboundedness and regular conditions. In *Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, FST TCS 2003, Proceedings*, volume 2914 of *Lecture Notes in Computer Science*, pages 88–99. Springer, 2003.

[14] J. Richard Büchi. State-strategies for games in $F_{\sigma\delta} \cap G_{\delta\sigma}$. *Journal of Symbolic Logic*, 48(4):295–311, 1983.

[15] J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:1171–1198, 1969.

[16] Thierry Cachat. Symbolic strategy synthesis for games on pushdown graphs. In *29th International Colloquium on Automata, Languages, and Programming, ICALP 2002, Proceedings*, volume 2380 of *Lecture Notes in Computer Science*, pages 704–715. Springer, 2002.

[17] Thierry Cachat. Two-way tree automata solving pushdown games. In E. Grädel, W. Thomas, and T. Wilke, editors, *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*, pages 303–317. Springer, 2002.

[18] Thierry Cachat. Uniform solution of parity games on prefix-recognizable graphs. In Antonin Kucera and Richard Mayr, editors, *4th International Workshop on Verification of Infinite-State Systems, Infinity 2002*, volume 68 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2002.

[19] Thierry Cachat. Higher order pushdown automata, the Caucal hierarchy of graphs and parity games. In *30th International Colloquium on Automata, Languages, and Programming, ICALP 2003, Proceedings*, volume 2719 of *Lecture Notes in Computer Science*, pages 556–569. Springer, 2003.

[20] Thierry Cachat. *Jeux sur des graphes d'automates à pile et leurs extensions*. PhD thesis, Université de Rennes 1, 2004.

[21] Thierry Cachat, Jacques Duparc, and Wolfgang Thomas. Solving pushdown games with a Σ_3 winning condition. In *11th Annual Conference of the European Association for Computer Science Logic, CSL 2002, Proceedings*, volume 2471 of *Lecture Notes in Computer Science*, pages 322–336. Springer, 2002.

[22] Didier Caucal. On infinite terms having a decidable monadic theory. In *Mathematical Foundations of Computer Science, 27th International Symposium, MFCS 2002, Proceedings*, volume 2420 of *Lecture Notes in Computer Science*, pages 165–176. Springer, 2002.

[23] Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, Marcin Jurdziński, and Freddy Mang. Interface compatibility checking for software modules. In *Computer Aided Verification, 14th International Conference, CAV 2002, Proceedings*, volume 2404 of *Lecture Notes in Computer Science*, pages 428–441. Springer, 2002.

[24] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the Association of Computing Machinery*, 28(1):114–133, 1981.

[25] Ashok K. Chandra and Larry J. Stockmeyer. Alternation. In *17th Annual Symposium on Foundations of Computer Science, FoCS 1976, Proceedings*, pages 98–108, Houston, Texas, 25–27 1976. IEEE.

[26] Rina S. Cohen and Arie Y. Gold. Theory of ω-languages. part one: characterizations of ω-context free languages. *Journal of Computer and System Sciences*, 15(2):169–184, 1979.

[27] Thomas Colcombet and Damian Niwiński. On the positional determinacy of edge-labeled games. Soumis, 2004.

[28] Luca de Alfaro. Quantitative verification and control via the mu-calculus. In *Concurrency Theory, 14th International Conference, CONCUR 2003, Proceedings*, volume 2761 of *Lecture Notes in Computer Science*, pages 103–127. Springer, 2003.

[29] Jacques Duparc. Wadge hierarchy and Veblen hierarchy. part I: Borel sets of finite rank. *Journal of Symbolic Logic*, 66(1):56–86, 2001.

[30] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, FoCS 1991, Proceedings*, pages 368–377. IEEE Computer Society Press, 1991.

[31] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model-checking for fragments of mu-calculus. In *Computer Aided Verification, 5th International Conference, CAV 1993, Proceedings*, volume 697 of *Lecture Notes in Computer Science*, pages 385–396. Springer, 1993.

[32] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model-checking for the mu-calculus and its fragments. *Theoretical Computer Science*, 258(1–2):491–522, 2001.

[33] Javier Esparza, David Hansel, Peter Rossmanith, and Stefan Schwoon. Efficient algorithms for model checking pushdown systems. In *Computer Aided Verification, 12th International Conference, CAV 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, pages 232–247. Springer, 2000.

[34] Kousha Etessami. Analysis of recursive game graphs using data flow equations. In Springer, editor, *Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004, Proceedings*, volume 2937 of *Lecture Notes in Computer Science*, pages 282–296, 2004.

[35] Olivier Finkel. Topological properties of omega context-free languages. *Theoretical Computer Science*, 262:669–697, 2001.

[36] Olivier Finkel. Borel hierarchy and omega context-free languages. *Theoretical Computer Science*, 290:1385–1405, 2003.

[37] Olivier Finkel. *Propriétés topologiques de quelques classes de langages de mots infinis*. Habilitation à diriger des recherches, Université Paris 7, France, 2003.

[38] Hugo Gimbert. Parity and exploration games on infinite graphs. In Springer, editor, *13th Annual Conference of the European Association for Computer Science Logic, CSL 2004, Proceedings*, Lecture Notes in Computer Science, 2004. A paraître.

[39] Hugo Gimbert and Wiesław Zielonka. When can you play positionally? In Springer, editor, *Mathematical Foundations of Computer Science, 29th International Symposium, MFCS 2004, Proceedings*, volume 3153 of *Lecture Notes in Computer Science*, pages 686–697, 2004.

[40] Erich Grädel. Positional determinacy on infinite games. In Springer, editor, *17th Annual Symposium on Theoretical Aspects of Computer Science, STACS 2004, Proceedings*, volume 2996 of *Lecture Notes in Computer Science*, pages 4–18, 2004.

[41] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[42] Erich Grädel and Igor Walukiewicz. Positional determinacy on infinite games. Soumis, 2004.

[43] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *14th Annual ACM Symposium on Theory of Computing, STOC 1982, Proceedings*, pages 60–65, 1982.

[44] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (2nd edition)*. Addison Wesley, 2001.

[45] Marcin Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Information Processing Letters*, 68(3):119–124, November 1998.

[46] Marcin Jurdziński. Small progress measures for solving parity games. In *17th Annual Symposium on Theoretical Aspects of Computer Science, STACS 2000, Proceedings*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2000.

[47] Alexander S. Kechris. *Classical Descriptive Set Theory*, volume 156 of *Graduate texts in mathematics*. Springer, 1994.

[48] Dexter Kozen. On parallelism in Turing machines. In *17th Annual Symposium on Foundations of Computer Science, FoCS 1976, Proceedings*, pages 89–97, Houston, Texas, 25–27 1976. IEEE.

[49] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. Extended temporal logic revisited. In *Concurrency Theory, 12th International Conference, CONCUR 2001, Proceedings*, volume 2154 of *Lecture Notes in Computer Science*, pages 519–535. Springer, 2000.

[50] Orna Kupferman and Moshe Y. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In *Computer Aided Verification, 12th International Conference, CAV 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, pages 36–52. Springer, 2000.

[51] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(3):408–429, 2001.

[52] Richard E. Ladner, Richard J. Lipton, and Larry J. Stockmeyer. Alternating pushdown automata (preliminary report). In *19th Annual Symposium on Foundations of Computer Science, FoCS 1978, Proceedings*, pages 92–106. IEEE, 1978.

[53] Matti Linna. On ω-sets associated with context-free languages. *Information and Control*, 1976.

[54] Christof Löding. Communication privée.

[55] Christof Löding. Methods for the transformation of ω-automata: Complexity and connection to second order logic. Diplomata thesis, Christian-Albrechts-University of Kiel, 1998.

[56] Christof Löding, P. Madhususdan, and Olivier Serre. Visibly pushdown games. In *Foundations of Software Technology and Theoretical Computer Science, 24th Conference, FST TCS 2004*, Lecture Notes in Computer Science. Springer, 2004. A paraître.

[57] Jerzy Marcinkowski and Tomasz Truderung. Optimal complexity bounds for positive LTL games. In *11th Annual Conference of the European Association for Computer Science Logic, CSL 2002, Proceedings*, volume 2471 of *Lecture Notes in Computer Science*, pages 262–275. Springer, 2002.

[58] Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(363–371), 1975.

[59] Richard Mayr. Strict lower bounds for model checking BPA. In *Electronic Notes in Theoretical Computer Science*, volume 18. Elsevier, 2000.

[60] Robert McNaughton. Finite-state infinite games. Technical report, Massachusetts Institute of Technology, September 1965.

[61] Robert McNaughton. Infinite games played on finite graphs. *Annals of pure and aapplied logic*, 65:149–184, 1993.

[62] Satoru Miyano and Takeshi Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

[63] Andrzej W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *5th Symposium on Computation Theory, Proceedings*, volume 208 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 1984.

[64] Andrzej W. Mostowski. Games with forbidden positions. Technical Report 78, University of Gdansk, 1991.

[65] David E. Muller and Paul E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985.

[66] Maurice Nivat. Mots infinis engendrés par une grammaire algébrique. *R.A.I.R.O. — Informatique Théorique et Applications*, 11:311–327, 1977.

[67] Maurice Nivat. Sur les ensembles de mots infinis engendrés par une grammaire algébrique. *R.A.I.R.O. — Informatique Théorique et Applications*, 12:259–278, 1978.

[68] Damian Niwiński. μ-calculus via games. In *11th Annual Conference of the European Association for Computer Science Logic, CSL 2002, Proceedings*, volume 2471 of *Lecture Notes in Computer Science*, pages 27–43. Springer, 2002.

[69] Christos H. Papadimitriou. *Complexity Theory*. Addison Wesley, 1994.

[70] Dominique Perrin and Jean-Eric Pin. *Infinite Words*. Academic Press, 2004.

[71] Nir Piterman. Extended temporal logic with ω-automata. Thesis for the M.Sc. degree, Weizmann Institute of Science, 2000.

[72] Nir Piterman and Moshe Vardi. Micro-macro stack systems: A new frontier of decidability for sequential systems. In *17th IEEE Symposium on Logic in Computer Science, LICS 2002, Proceedings*, pages 381–390. IEEE Computer Society, 2002.

[73] Jacques Sakarovitch. *Elements de théorie des automates*. Vuibert, 2003.

[74] Olivier Serre. Games with winning conditions of high Borel complexity. *Theoretical Computer Science*. à paraître.

[75] Olivier Serre. Note on winning positions on pushdown games with ω-regular conditions. *Information Processing Letters*, 85:285–291, 2003.

[76] Olivier Serre. Games with winning conditions of high Borel complexity. In *31st International Colloquium on Automata, Languages, and Programming, ICALP 2004, Proceedings*, volume 3142 of *Lecture Notes in Computer Science*, pages 1150–1162. Springer, 2004.

[77] Wolfgang Thomas. On the synthesis of strategies in infinite games. In *8th Annual Symposium on Theoretical Aspects of Computer Science, STACS 1995, Proceedings*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 1995.

[78] Wolfgang Thomas. Infinite games and verification (extended abstract of a tutorial). In *Computer Aided Verification, 14th International Conference, CAV 2002, Proceedings*, volume 2404 of *Lecture Notes in Computer Science*, pages 58–64. Springer, 2002.

[79] Moshe Y. Vardi. Reasoning about the past with two-way automata. In *25th International Colloquium on Automata, Languages, and Programming, ICALP 1998, Proceedings*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer, 1998.

[80] Jens Vöge and Marcin Jurdziński. A discrete strategy improvement algorithm for solving parity games (Extended abstract). In *Computer Aided Verification, 12th International Conference, CAV 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, pages 202–215. Springer, 2000.

[81] William W. Wadge. *Reducibility and determinateness of the Baire space*. PhD thesis, Berkeley, 1984.

[82] Klaus Wagner and Gerd Wechsung. *Computational complexity*. D. Reidel Publishing Company, 1986.

[83] Igor Walukiewicz. Pushdown processes: games and model checking. In *Computer Aided Verification, 8th International Conference, CAV 1996, Proceedings*, volume 1102 of *Lecture Notes in Computer Science*, pages 62–74. Springer, 1996.

[84] Igor Walukiewicz. Pushdown processes: games and model checking. *Information and Computation*, 157:234–263, 2000.

[85] Igor Walukiewicz. A landscape with games in the background. In *19th IEEE Symposium on Logic in Computer Science, LICS 2004, Proceedings*, pages 356–366. IEEE Computer Society, 2004.

[86] Thomas Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bulletin of the Belgian Mathematical Society*, 8(2):359–391, 2001.

[87] Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.