Contributions à la génération aléatoire pour des classes d'automates finis

Jean-Luc Joly

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UFC UNIVERSITÉ DE FRANCHE-COMTÉ

THÈSE présentée par

JEAN-LUC JOLY

pour obtenir le

Grade de Docteur de
l'Université de Franche-Comté

Spécialité : **Informatique**

Contributions à la génération aléatoire uniforme pour des classes d'automates finis

Soutenue publiquement le 23 mars 2016 devant le Jury composé de :

JULIEN CLÉMENT	Rapporteur du jury	Chargé de Recherche CNRS HDR, Université de Caen
CYRIL NICAUD	Rapporteur du jury	Professeur des Universités, Université Paris-Est, Marne la Vallée
JEAN-FRANÇOIS COUCHOT	Membre du jury	Maître de Conférences, Université de Franche-Comté
JULIEN DAVID	Membre du jury	Maître de Conférences, Université Paris 13
PIERRE-CYRILLE HÉAM	Directeur de Thèse	Professeur des Universités, Université de Franche-Comté
OLGA KOUCHNARENKO	Membre du jury	Professeur des Universités, Université de Franche-Comté

Remerciements

Les travaux qui suivent sont ceux d'un vieil étudiant, lancé dans l'informatique grâce au Centre de Télé-enseignement Universitaire de l'université de Franche-Comté (CTU).

L'évolution et le développement de ce merveilleux outil permet aujourd'hui de poursuivre des études en informatique, jusqu'au niveau Master 2, tout en menant une vie professionnelle et familiale. Je remercie l'université de Franche-Comté et en particulier Isabelle Jacques d'avoir su construire et organiser ce dispositif aujourd'hui parfaitement huilé !

Les années passées au CTU ont supposé de la ténacité et du soutien. Toute ma sympathie va à Anne Alvarez et Christophe Enderlin qui m'ont accompagné de la Licence 3 jusqu'au Master 2. Nos discussions et nos échanges ont toujours été enrichissants et un moyen de rebondir lors des périodes de doute.

L'idée de poursuivre en thèse a mûri grâce aux conseils avisés d'amis bisontins de l'université, Bénédicte Hermann et Laurent Philippe. Je les remercie pour le temps qu'ils m'ont consacré !

L'homme clef de l'aboutissement de ces travaux de thèse reste cependant Pierre-Cyrille Héam. Après avoir encadré mon stage de Master, je savais pouvoir lui faire une confiance absolue. Il a été un directeur attentif, d'une disponibilité sans faille, à l'écoute des motivations et des contraintes de son étudiant. Si j'ai beaucoup appris scientifiquement à son contact, je pense aussi avoir fait quelques progrès en m'inspirant de sa patience et de sa sérénité dans ces situations où les agendas professionnels et familiaux s'entrechoquent un peu violemment.

Pierre-Cyrille Héam m'a par ailleurs fait confiance pour représenter le laboratoire au congrès CIAA 2015, qui se tenait en Suède, pour y exposer nos travaux communs. Merci pour cette confiance et cette semaine importante de rencontres et d'échanges.

Il fallait finalement une bonne dose de courage pour accepter de superviser les travaux d'un enseignant de classe préparatoire à plus de 500 km du laboratoire ! Je remercie vivement Pierre-Cyrille d'avoir été dans ce contexte, un peu compliqué, l'encadrant idéal !

Je remercie beaucoup les rapporteurs de cette thèse, Julien Clément et Cyril Nicaud d'avoir accepter de bonne grâce cette tâche difficile et prenante. Leurs commentaires montrent qu'ils ont réalisé une lecture en profondeur de ce travail. Je remercie Julien Clément pour sa grande minutie et les suggestions qu'il a bien voulu me faire. Je remercie également Cyril Nicaud de m'avoir consacré une journée complète à Marne-la-Vallée. Au cours de cette rencontre, j'ai fait moisson de belles idées qui auraient mérité sans doute des développements plus à la hauteur.

C'est avec un très grand plaisir que j'ai pris connaissance de la composition du jury. Olga Kouchnarenko a été la première personne à me parler d'automates, je suis très heureux et flatté qu'elle soit à l'arrivée d'un processus dont elle a été une initiatrice. Les travaux de Julien David m'ont souvent accompagné et ont toujours enrichi ma réflexion. Je le remercie ainsi que Jean-François Couchot d'honorer le jury de leur présence.

Le contexte particulier de cette thèse promettait d'être aventureux. Une réforme des programmes de mathématiques, l'introduction de l'informatique puis l'ouverture d'une nouvelle option, dans la classe dont j'ai la charge, n'ont fait que pimenter les choses. Il a fallu apprendre à travailler partout et en toutes circonstances. Je remercie par exemple le foyer du conservatoire d'Aix en Provence de m'avoir servi de lieu de travail très agréable pendant que le petit dernier soufflait dans son instrument...

La charge de travail supplémentaire représentée par ces années d'études a pesé sur la famille. Je ne serais pas arrivé à mes fins sans le soutien d'Isabelle et des enfants. Ils ont supporté mes humeurs et mes manquements. Leur mérite ne se mesure pas. Merci à vous !

Résumé

Le concept d'automate, central en théorie des langages, est l'outil d'appréhension naturel et efficace de nombreux problèmes concrets. L'usage intensif des automates finis dans un cadre algorithmique s'illustre par de nombreux travaux de recherche. La correction et l'évaluation sont les deux questions fondamentales de l'algorithmique. Une méthode classique d'évaluation s'appuie sur la génération aléatoire contrôlée d'instances d'entrée. Les travaux décrits dans cette thèse s'inscrivent dans ce cadre et plus particulièrement dans le domaine de la génération aléatoire uniforme d'automates finis.

L'exposé qui suit propose d'abord la construction d'un générateur aléatoire d'automates à pile déterministes, *real time*. Cette construction s'appuie sur la méthode symbolique. Des résultats théoriques et une étude expérimentale sont exposés.

Un générateur aléatoire d'automates non-déterministes illustre ensuite la souplesse d'utilisation de la méthode de Monte-Carlo par Chaînes de Markov (MCMC) ainsi que la mise en œuvre de l'algorithme de Metropolis-Hastings pour l'échantillonnage à isomorphisme près. Un résultat sur le temps de mélange est donné dans le cadre général.

L'échantillonnage par méthode MCMC pose le problème de l'évaluation du temps de mélange dans la chaîne. En s'inspirant de travaux antérieurs pour construire un générateur d'automates partiellement ordonnés, on montre comment différents outils statistiques permettent de s'attaquer à ce problème.

Abstract

The concept of automata, central to language theory, is the natural and efficient tool to apprehend various practical problems.

The intensive use of finite automata in an algorithmic framework is illustrated by numerous research works.

The correctness and the evaluation of performance are the two fundamental issues of algorithmics.

A classic method to evaluate an algorithm is based on the controlled random generation of inputs.

The work described in this thesis lies within this context and more specifically in the field of the uniform random generation of finite automata.

The following presentation first proposes to design a deterministic, real time, pushdown automata generator. This design builds on the symbolic method. Theoretical results and an experimental study are given.

A random generator of non deterministic automata then illustrates the flexibility of the Markov Chain Monte Carlo methods (MCMC) as well as the implementation of the Metropolis-Hastings algorithm to sample up to isomorphism. A result about the mixing time in the general framework is given.

The MCMC sampling methods raise the problem of the mixing time in the chain. By drawing on works already completed to design a random generator of partially ordered automata, this work shows how various statistical tools can form a basis to address this issue.

Sommaire

Introduction

Cadre général

Historiquement, la théorie des langages est d'abord le champ d'étude naturel des linguistes. Les sciences informatiques se sont très vite approprié les objets et les concepts de cette théorie : les *mots*, les *langages formels*, leur *classification*. Cette phase d'appropriation s'est naturellement poursuivie par une phase de développement. La théorie des langages est ainsi devenue une branche essentielle de l'informatique fondamentale théorique aussi bien qu'appliquée. La frontière est d'ailleurs parfois assez floue entre les champs théoriques et applicatifs. Il est néanmoins clair que les concepts du domaine sont devenus fondamentaux dans les sciences numériques. Ils se déploient notamment dans les domaines de la calculabilité, de l'algorithmique du traitement de texte, de la compilation, de la vérification, du codage, de l'apprentissage automatique. On peut aussi remarquer avec intérêt que la théorie des langages, en informatique, réalise souvent aujourd'hui un retour aux sources de la linguistique. En effet, des questions aux enjeux essentiels comme le traitement des requêtes dans un moteur de recherche, la reconnaissance vocale ou la traduction automatique d'une langue naturelle vers une autre sont devenus des champs de recherche cruciaux en informatique. Les travaux exposés dans [DBL13] sont une des illustrations exemplaires des liens profonds entre informatique théorique et langages naturels [1].

Les automates finis, sous différentes déclinaisons, sont les objets centraux de cette thèse. Directement inspirés des machines de Turing, ils constituent des modèles aussi simples qu'efficaces de calcul ou de reconnaissabilité. Ils sont les outils incontournables de l'étude de certaines familles de langages. Ainsi, la famille des langages reconnus par automates finis, dans leur définition la plus "simple", est exactement la famille des langages réguliers. Ce résultat est bien connu, il faut néanmoins le prolonger et l'entendre en comprenant qu'il garantit que le concept d'automate est l'outil d'appréhension naturel et efficace de nombreux problèmes concrets. Vus comme graphes étiquetés, les automates finis s'utilisent avec une bonne partie de l'arsenal théorique et algorithmique accompagnant les graphes (les algorithmes de parcours sont par exemple très facilement adaptables aux automates). De façon plus générale, la richesse des propriétés mathématiques des automates ou des langages réguliers permet d'établir des connexions remarquables avec la logique, l'algèbre et même l'arithmétique. Ces liens profonds permettent la construction de théories solides et le développement d'algorithmes fins.

L'usage intensif des automates finis sur le plan algorithmique s'illustre par de nombreux travaux de recherche. Comme souvent, il s'agit d'optimiser des procédures existantes ou de développer des idées originales plus efficaces. Le test de l'inclusion de deux langages réguliers définis par automates finis est un exemple qui illustre le propos. Ce problème, connu pour être PSPACE-complet [Alb72], fait l'objet de plusieurs études récentes [2], par exemple [Par10]. Les investigations sur cette question sont notamment motivées par ses applications dans le domaine du *model-checking*. Il ne s'agit là que d'un exemple parmi

1. Cette référence n'épuise évidemment pas le très vaste sujet des liens évoqués entre informatique théorique et langages naturels.
2. http://www.languageinclusion.org/doku.php

d'autres : l'apprentissage automatique de langages réguliers, le calcul du complémentaire d'un langage régulier de mots infinis, sont aussi au cœur de nombreuses études d'optimisations algorithmiques s'appuyant sur des automates finis.

Motivations

Si les objets de ce travail sont les automates, ses motivations sont essentiellement liées à l'évaluation des algorithmes. Le développement d'un nouvel algorithme passe en effet toujours par le problème de l'évaluation de son efficacité. Cette question a très souvent un versant théorique et un versant pratique et plusieurs déclinaisons possibles qui ne répondent pas toujours exactement au même problème. Souvent, les études portent sur l'analyse de la complexité dans le pire des cas. Cette étape, bien que nécessaire, n'est cependant pas toujours significative. Les cas les pires peuvent en effet s'avérer exceptionnels en pratique. L'exemple le plus connu illustrant ce propos est celui du Quick-sort, plus efficace en pratique que le Heap-sort mais avec une complexité dans le cas le pire moins bonne. L'évaluation de la complexité en moyenne permet de lisser les résultats en évitant les effets de loupe sur les cas exceptionnels. La complexité en moyenne est donc, en ce sens, une bonne mesure de l'efficacité d'un algorithme, ou au moins une mesure qui complète utilement la complexité dans le cas le pire (le cas moyen n'étant pas nécessairement le cas usuel pratique) .

Le calcul de la complexité en moyenne est cependant un exercice théorique beaucoup plus difficile que l'analyse dans le cas le pire et cette étude est rarement faite sur les algorithmes. Dans le cadre de la théorie des automates et des expressions régulières, les résultats de [BDN12, Jul12], par exemple, illustrent bien l'apport de l'étude en moyenne : l'algorithme de minimisation de Moore, plus mauvais dans le cas le pire que celui de Hopcroft, a un *meilleur comportement* en moyenne dans le sens où l'impression pratique est celle d'une meilleure performance de l'algorithme de Moore.
L'algorithme de Hopcroft n'est pas déterministe et même en étant asymptotiquement plus efficace que l'algorithme de Moore, son implémentation pratique peut s'avérer plus délicate et conduire, de part la lourdeur des structures de données utilisées, à des opérations plus coûteuses. Ces questions sont en particulier développées dans [Jul12].

La validité de la mesure est cependant inhérente au choix de la distribution sur les données d'entrée. Ce choix est connu pour être une question très délicate. Des travaux s'intéressant aux enjeux liés à ce choix, dans le cas des automates finis, ont été réalisés [NPR10]. Ils concernent souvent des spécialistes de l'analyse en moyenne qui s'attaquent à des résultats difficiles à obtenir. Il faut d'ailleurs noter que les questions deviennent plus ardues encore lorsque les algorithmes comportent des optimisations.

L'évaluation empirique de performances est un autre angle d'attaque de la question. C'est un moyen accessible qui peut être considéré comme complémentaire des autres approches. La facilité relative de mise en œuvre de l'approche empirique se paie par l'absence de preuve formelle des résultats et par l'inconfort intellectuel lié à l'aspect *boite noire* du procédé.

On peut distinguer d'autres approches d'évaluation des performances d'algorithmes. Ici encore, il ne s'agit pas de choisir une voie plutôt qu'une autre, mais de multiplier des outils complémentaires qui permettent d'affiner la perception générale [3] :

- L'utilisation de *benchmarks* consiste à s'appuyer sur une base de données d'exemples concrets des entrées possibles de l'algorithme. L'avantage de cette approche réside clairement dans la validation (ou l'invalidation !) de l'algorithme par son évaluation sur des cas réels concrets d'utilisation. Ses revers sont de deux sortes :
  - Le développement de l'algorithme peut, même involontairement, être piloté par la liste des données d'entrée. Cette liste n'est jamais exhaustive et le processus

3. Notons que les trois approches décrites ci-dessous sont utilisées dans les compétitions de SAT-solvers : http://www.satcompetition.org/SATsolvers

de développement peut évoluer vers un algorithme qui n'optimise que la liste des entrées de la *benchmark*.
  - Dans de nombreux cas l'élaboration de *benchmarks* cohérentes, valides et consensuelles est tout simplement impossible et la méthode est alors inapplicable.
- La génération aléatoire d'instances consiste à construire un échantillonneur de celles-ci. Cette approche permet une bonne estimation du comportement moyen d'un algorithme. Une des questions centrales de la méthode réside dans le choix d'une distribution pertinente au regard de l'utilisation envisagée de l'algorithme. Le plus souvent, le choix se porte sur une distribution uniforme sur l'ensemble de toutes les instances ou sur une sous-classe. La difficulté de la mise en œuvre de la méthode est souvent liée à la nature des objets à engendrer. Ils sont à la fois trop nombreux et structurellement trop complexes pour en faire une liste exhaustive qui permettrait de ramener le problème de l'échantillonnage à un simple générateur aléatoire, pseudo-uniforme, de nombres entiers.
- L'utilisation d'instances connues difficiles, éventuellement tirées au sort, permet d'estimer le comportement de l'algorithme dans les situations les pires. Comme pour les *benchmarks*, il est assez difficile d'avoir accès à de tels instances et encore plus de savoir les tirer au sort.

État de l'art sur la génération aléatoire d'automates

Malgré quelques résultats anciens sur le dénombrement des automates [Kor78, Kor86], les travaux sur la génération aléatoire sont assez récents.

1. La première approche pour engendrer aléatoirement et uniformément des automates déterministes, accessibles et complets consiste à utiliser une bijection vers une structure combinatoire décomposable. C'est la méthode développée dans [Nic00] pour les alphabets à deux lettres. On trouve dans [CP05] un prolongement des mêmes idées pour tout alphabet. Une autre décomposition des automates déterministes, accessibles, est proposée dans [AMR07]. Ces travaux, s'appuyant sur la méthode symbolique, permettent une génération en $O(n^3)$ (où n est le nombre d'états). Plus récemment, la méthode récursive a aussi été utilisée dans [DFN13] pour engendrer uniformément des automates acycliques accessibles et déterministes.

2. L'utilisation de générateurs de Boltzmann a permis d'augmenter très significativement la performance de la génération, en proposant des algorithmes de complexité $O(n^{\frac{3}{2}})$ [BN07, BDN09].

3. L'utilisation d'algorithmes à rejet permet aussi une génération très efficace (et très simple) [CN12]. Notons qu'on trouve aussi dans [Nic00] une méthode à rejet s'appuyant sur la combinatoire des permutations pour engendrer en $O(n)$ des automates à groupe.

4. Toujours dans le cadre des automates déterministes accessibles, les travaux [CF11, CF12] ont ouvert la voie à l'emploi de générateurs de type Monte-Carlo, en traitant le cas des automates acycliques.

5. La génération aléatoire d'automates non-déterministes est un sujet moins traité et les techniques manquent souvent de fondement théorique. Le modèle le plus utilisé est probablement celui de [TV05] qui s'appuie sur des modèles de type Erdös-Rényi. On peut aussi citer les travaux de [CHPZ02] qui s'appuient sur l'utilisation de mots aléatoires.

Le récent survey [Nic14] dresse un panorama complet de la génération aléatoires d'automates déterministes.

Contributions

Cette thèse, propose les contributions suivantes :

1. Les automates à pile, déterministes, accessibles et *real-time*, c'est-à-dire sans ε-transitions, constituent notre premier sujet d'étude. Nous avons utilisé les méthodes d'analyse combinatoire classiques pour ce problème.
   - Nous nous appuyons sur des travaux récents [BN07] afin de d'engendrer l'automate fini sous-jacent. Nous élaborons naturellement un générateur pour engendrer les étiquettes des transitions. On obtient ainsi un générateur aléatoire uniforme.
   - Nous donnons, dans le cas des automates complets, une évaluation asymptotique du nombre de tels automates.
   - Nous avons effectué une étude expérimentale qualitative des automates engendrés, notamment en fonction du mode de reconnaissance.
2. Notre seconde contribution s'appuie, quant à elle, sur la génération par chaîne de Markov, afin de produire des automates non-déterministes.
   - La méthode s'applique à plusieurs classes d'automates non-déterministes. Dans le cas le plus général, sans contrainte d'accessibilité, nous donnons une majoration du temps de mélange. Elle est polynomiale sur le nombre d'états des automates pour la chaîne construite. Les déplacements dans la chaîne de Markov se font eux aussi en temps polynomial.
   - Nous montrons ensuite comment utiliser l'algorithme de Metropolis-Hastings [CG95, CC96] afin d'obtenir une génération à isomorphisme près. On montre que l'algorithme est applicable si l'on sait calculer le nombre d'automorphismes d'un automate donné.
   - Nous montrons que le problème de calcul du nombre d'automorphismes peut, théoriquement, se résoudre en temps polynomial si l'on borne le degré sortant des automates considérés.
   - En pratique, on montre qu'en utilisant des étiquetages (*labelings*), technique classique de l'algorithmique des graphes, ont peut très efficacement calculer la taille du groupe d'automorphismes d'un automate ayant plusieurs centaines d'états.
   - L'implémentation nous permet d'engendrer relativement efficacement, à isomorphisme près, des automates non-déterministes ayant plusieurs dizaines d'états. Cependant, dans les cas intéressants, nous ne savons pas estimer le temps de mélange.
3. Notre troisième contribution porte sur l'utilisation de tests statistiques, sur le temps de mélange, dans le cadre de la génération d'automates finis. En s'appuyant fortement sur les travaux de [CF12], nous montrons comment engendrer par chaînes de Markov des automates déterministes, accessibles et partiellement ordonnés. Nous avons mis en place plusieurs tests statistiques afin d'évaluer un possible temps de mélange. L'idée principale de cette section est de montrer que des tests simples, comme les tests d'auto-corrélation ou de Gelman-Rubin, sont très faciles à mettre en place. Ces tests permettent, lorsqu'ils sont négatifs, d'invalider une hypothèse à propos du temps de mélange. Comme souvent en statistiques, il est difficile d'obtenir une réponse catégoriquement positive. On avance le plus souvent à partir d'une réponse du type : "on ne peut pas dire que notre hypothèse est fausse". On montre donc comment prolonger les tests en s'appuyant sur une partition de la classe combinatoire permettant de faire un test du χ^2. Le générateur aléatoire basé sur une méthode de type Monte-Carlo par Chaînes de Markov (abrégé en MCMC) engendre des automates partiellement ordonnés. Les résultats obtenus suggèrent que le temps de mélange est de $O\left(n^3\right)$ où n est le nombre d'états des automates. L'idée très simple qui consiste à obtenir une partition de la classe des objets engen-

drés est également intéressante du point de vue de la combinatoire. En effet, tous les générateurs aléatoires considérés dans ce travail concernent des ensembles dont le cardinal interdit une génération exhaustive. Valider le caractère uniforme de ces générateurs en étudiant un échantillon de taille 10^5 ou 10^9 alors que le cardinal de la classe étudiée est de l'ordre de 10^{40} suppose la mise en place de stratégies de ce type.

Plan

Cette thèse est découpée en deux parties.

La première partie est consacrée aux préliminaires théoriques. Les notations, les notions et les résultats utilisés y sont explicités. Cette partie est elle même organisée en trois chapitres.

Le premier chapitre, **Automates**, traite des automates finis. Dans ce travail, les automates finis apparaissent sous trois formes :

- Les automates de mots. Il s'agit des automates finis dans le sens le plus originel du terme : les reconnaisseurs des langages réguliers.
- Les automates à pile. On rappelle quelques notions, en particulier celle d'automate *real-time* et les liens avec les langages hors-contextes.
- Les automates partiellement ordonnés. Ils constituent la dernière déclinaison des automates pour lesquels des générateurs aléatoires ont été mis au point dans ce travail. Ces automates sont moins classiques.

Le deuxième chapitre, **Analyse combinatoire, génération aléatoire**, est l'occasion de quelques rappels. L'objectif est de montrer comment les principes d'analyse combinatoire conduisent à deux façons d'envisager la construction de générateurs aléatoires par la méthode récursive d'une part, par les générateurs de Boltzmann ensuite. Les deux approches passent par la méthode symbolique qui donne lieu à des rappels élémentaires.

Le troisième chapitre, **Chaînes de Markov, génération aléatoire, statistiques**, regroupe tous les aspects *stochastiques* de cette thèse. Le but est, au delà des résultats théoriques, de faire appréhender le principe de la méthode en s'appuyant sur des exemples aussi simples que possibles. Le ton didactique de la section est une volonté du rédacteur ! La partie du troisième chapitre consacrée aux statistiques est suffisamment étoffée pour donner quelques clefs d'un domaine utilisé de façon moins commune que les précédents en informatique. Il nous a semblé en effet que la question méritait plus que *quelques rappels sur les résultats usuels* du domaine. Si les outils statistiques veulent se faire une place dans les sciences informatiques il faut sans doute leur accorder l'éclairage théorique qu'ils méritent.

La seconde partie est consacrée à la présentation des contributions présentées ci-avant. Elle est aussi organisées en trois chapitres :

- **Génération aléatoire d'automates à pile**, en utilisant la méthode symbolique.
- **Génération aléatoire, à isomorphisme près, d'automates non déterministes**, en utilisant des chaînes de Markov,
- **Génération aléatoire d'automates déterministes, accessibles et partiellement ordonnés : évaluation statistique du temps de mélange**

Les travaux effectués dans cette thèse ont donné lieu aux deux publications suivantes :

1. P.-C. Héam et J.-L. Joly, On the Uniform Random Generation of Non-deterministic Automata up to isomorphism, in Proceedings of the 20th International Conference on Implementation and Application of Automata (CIAA'15), (2015) LNCS.
2. P.-C. Héam et J.-L. Joly, Random Generation and Enumeration of Accessible Deterministic Real-time Pushdown Automata, in Proceedings of the 20th International Conference on Implementation and Application of Automata (CIAA'15), (2015) LNCS.

Première partie

Notions préliminaires

Chapitre 1

Automates

Sommaire

1.1 Automates

Les notions d'*alphabet, lettre, mot, langage, produit de mots* ne sont pas redéfinies. Une référence très complète pour la théorie des automates est [Sak09].

1.1.1 Quelques rappels

Les automates sont des machines très simples et familières qui représentent un des moyens de définir les langages rationnels [1]. Quelques points aussi essentiels qu'élémentaires sont rappelés ici. Certaines de ces propriétés, comme le fait que la classe des langages reconnus ne dépende pas du caractère déterministe ou non des automates considérés, sont évoquées surtout pour souligner la différence avec les automates à pile pour lesquelles ces distinctions sont importantes.

Définition 1 (Automate)**.** *Un automate $\mathcal{A}$, sur un alphabet Σ, est un 5-uplet défini par $\mathcal{A} \stackrel{def}{=} (Q, \Sigma, \delta, I, F)$ où :*

- *Q est un ensemble **fini** (l'ensemble des états),*
- *Σ est un alphabet,*
- *δ est une relation 3-aire sur les ensembles Q, Σ, Q, les éléments de δ sont appelés les transitons de $\mathcal{A}$,*
- *I est un sous-ensemble de Q constitué des états initiaux,*
- *F est un sous ensemble de Q constitué des états finals.*

1. On dit encore : langages réguliers

Remarque.
Avec les mêmes notations, on parle d'automate *avec ε-transitions* lorsque δ est une relation 3-aire sur les ensembles Q, $\Sigma \cup \{\varepsilon\}$ et Q. Où ε désigne le mot vide.

On note souvent un élément (p, a, q) de δ sous la forme $p \xrightarrow{a} q$ pour signifier que l'on passe de l'état p à l'état q en lisant la lettre a sur le mot d'entrée. Un automate, comme le montre la figure 1.1, est classiquement représenté par un graphe dont les sommets sont éventuellement signalés comme correspondant à un état initial ou final alors que les arêtes sont étiquetées par une lettre de l'alphabet Σ.

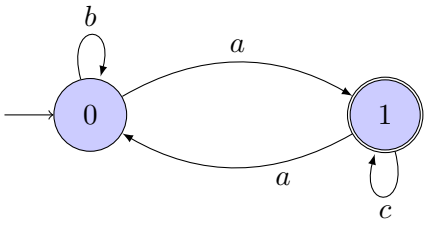


FIGURE 1.1 – Automate (fini), exemple

Un chemin dans un automate $\mathcal{A}$ est une suite finie de transitions consécutives

$$(q_0, a_1, q_1), \ldots, (q_{n-1}, a_n, a_n),$$

que l'on note aussi :

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} q_n.$$

Dans ce chemin, q_0 est l'*état de départ*, q_n l'*état d'arrivée* et ω l'*étiquette*. On dit que n, le nombre de transitions du chemin, est la *longueur* du chemin. Le chemin est dit *acceptant* lorsque $q_0 \in I$ et $q_n \in F$. Un mot ω sur l'alphabet Σ est *accepté* par $\mathcal{A}$ s'il est l'étiquette d'un chemin acceptant. L'ensemble des mots acceptés par $\mathcal{A}$ est le langage reconnu par $\mathcal{A}$, on le note $\mathcal{L}(\mathcal{A})$. On note parfois qu'il existe un chemin de q_0 à q_n étiqueté par ω par $q_0 \xrightarrow{\omega} q_n$.

Un automate est dit *accessible* si pour tout état q il existe un chemin d'un état initial p à l'état q. On dit que l'automate est *co-accessible* si pour tout état q il existe un chemin de q à un état final.

Un automate est dit *trim* ou *émondé* s'il est à la fois accessible et co-accessible.

Un automate est *complet* lorsque pour tout couple $(p, a) \in Q \times \Sigma$ il existe $q \in Q$ tel que le triplet (p, a, q) appartienne à l'ensemble δ.

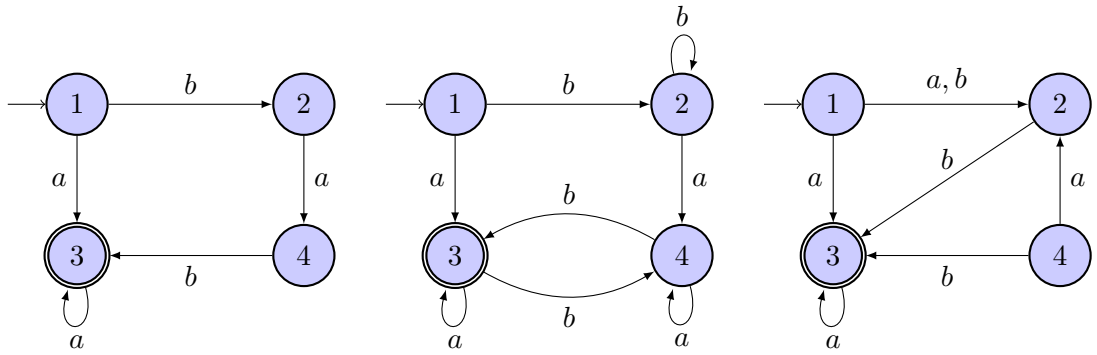
Un automate est *déterministe* lorsqu'il n'admet qu'un seul état initial ($|I| = 1$) et que la relation δ, vue comme relation binaire de $Q \times \Sigma$ sur Q est fonctionnelle : pour tout $(p, a) \in Q \times \Sigma$ il existe au plus un état q tel que $(p, a, q) \in \delta$. La figure1.2 donne des exemples pour ces notions.

Remarques.

- Si l'automate $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ est déterministe, il existe alors $q_0 \in Q$ tel que $I = \{q_0\}$. On écrit alors souvent $(Q, \Sigma, \delta, q_0, F)$ à la place de $(Q, \Sigma, \delta, \{q_0\}, F)$.
- Si l'automate $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ est déterministe, on parle souvent de la *fonction* de transition δ à la place de la *relation* δ. Si $(p, a, q) \in \delta$, on écrit couramment $\delta(p, a) = q$.

Si l'automate est par ailleurs complet, δ peut être vue comme une *application* ou une *fonction totale* de $Q \times \Sigma$ dans Q.

La figure 1.2 illustre ces notions.



automate déterministe — automate complet déterministe — automate non-déterministe

FIGURE 1.2 – Automate déterministe, automate complet déterministe, automate non-déterministe

On distingue aussi les automates standards et normalisés :

On dit d'un automate $\mathcal{A}$ qu'il est *standard* si et seulement si $\mathcal{A}$ admet un seul état initial et si l'état initial n'est l'état d'arrivée d'aucune transition.

On dit d'un automate $\mathcal{A}$ qu'il est *normalisé* si et seulement si $\mathcal{A}$ admet un seul état initial et un seul état final et par ailleurs si l'état initial n'est l'état d'arrivée d'aucune transition et que l'état final n'est l'état de départ d'aucune transition.
Un automate normalisé est standard. La réciproque est fausse.

Remarques.

- L'automate de la figure 1.1, n'est pas standard. En effet, si cet automate admet bien un seul état initial, celui-ci est en revanche l'état d'arrivée d'une transition.
- Les trois automates de la figure 1.2 sont standards. Ils n'ont qu'un seul état initial qui n'admet que des transitions sortantes. Ils ne sont en revanche pas normalisés dans la mesure où l'état final admet une transition sortante. La figure 1.3 représente un automate normalisé équivalent (dans le sens où il reconnaît le même langage, au premier automate de la figure 1.2.

Remarque.
On note que l'usage de ε-transitions a permis de normaliser aisément le premier automate de la figure 1.2.

1.1.2 Quelques résultats classiques

Automates et langages rationnels

On définit sur l'ensemble des langages sur Σ deux opérations binaires : l'addition et le produit. On définit aussi une opération unaire : l'étoile.

- L'addition $L + L'$ des deux langages L et L' est définie comme l'union de L et L'.
- Le produit $L \cdot L'$ des deux langages L et L' est défini comme l'ensemble des mots de la forme $\omega\omega'$, concaténation d'un mot ω de L et d'un mot ω' de L'.

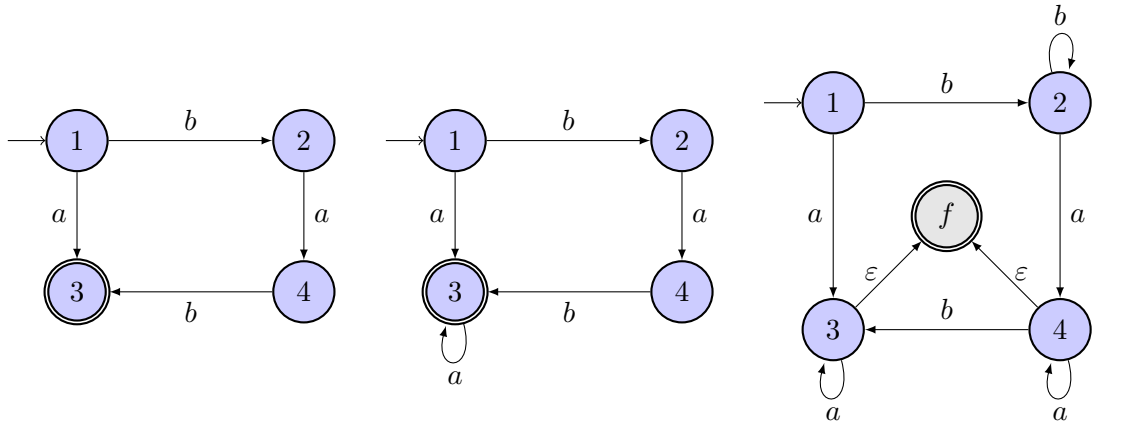


FIGURE 1.3 – Automate standard et automate normalisé équivalent

- L'étoile de L, L^* est définie comme l'union des langages L^n, où L^n est défini inductivement par :
$$L^0 = \{\varepsilon\}, \quad L^{n+1} = L \cdot L^n.$$
où ε désigne le mot vide.

Les langages rationnels sur l'alphabet Σ sont définis comme la plus petite classe $\mathcal{R}$ des langages satisfaisant :

- $\emptyset \in \mathcal{R}$
- Pour tout $a \in \Sigma$, le singleton $\{a\}$ appartient à $\mathcal{R}$.
- $\mathcal{R}$ est stable par les opérations d'addition, de multiplication et l'étoile.
C'est-à-dire que si L et L' sont deux langages de $\mathcal{R}$, alors $L + L'$, $L \cdot L'$, L^* sont encore des langages de $\mathcal{R}$.

On ne développera pas ici la notion d'expression rationnelle qui naît directement de cette définition.
Les langages rationnels peuvent aussi être définis par les grammaires du même nom. Nous ne développerons pas non plus cet aspect des choses.

Le résultat essentiel, connu sous le nom de Théorème de Kleene est le suivant :

Théorème 2 (Langages rationnels et automate)**.** *Un langage L est rationnel si et seulement s'il existe un automate $\mathcal{A}$ tel que $L = \mathcal{L}(\mathcal{A})$.*

Déterminisme et complétude

On peut se demander quels liens existent entre les différentes familles d'automates. Le déterminisme impose des contraintes fortes sur l'ensemble des transitions, δ doit en effet pouvoir être vue comme une fonction sur $Q \times \Sigma$.

De nombreux problèmes (comme l'inclusion) ont des complexités meilleures sur les automates déterministes que sur les automates en général.

Proposition 3 (Déterminisme et complétude)**.** *Soit L un langage. Les propositions suivantes sont équivalentes.*

- *L est reconnaissable par un automate,*
- *L est reconnaissable par un automate déterministe,*
- *L est reconnaissable par un automate complet,*
- *L est reconnaissable par un automate déterministe et complet.*

Il existe une méthode constructive pour passer d'un automate à un automate déterministe reconnaissant le même langage. On rappelle que "le prix à payer" pour cette déterminisation est l'explosion du nombre d'états puisque l'on peut très bien passer d'un automate à n états à un automate à 2^n états. Rendre un automate complet est en revanche une opération très simple qui consiste, au plus, à ajouter un état.

1.1.3 Automates isomorphes

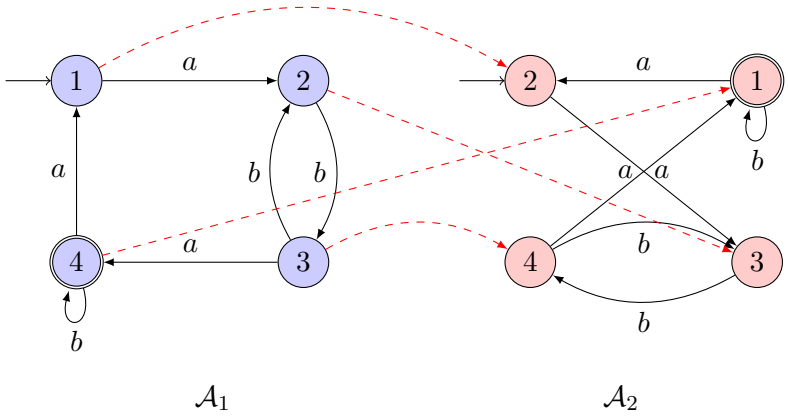
D'un point de vue sémantique, et dans le cadre de très nombreuses applications des automates, le nom des états n'est pas important. Ce qui importe, c'est les transitions entres les états et les étiquettes de ces transitions. Il est donc utile, notamment pour la génération aléatoire, de définir une relation d'équivalence entre deux automates. La relation d'équivalence est une relation d'égalité au nom des états près. C'est la notion d'isomorphisme qui permet de la définir formellement.

Définition 4 (Automates isomorphes)**.** *On considère* $\mathcal{A}_1 = (Q_1, \Sigma, \Delta_1, I_1, F_1)$ *et* $\mathcal{A}_2 = (Q_2, \Sigma, \Delta_2, I_2, F_2)$ *deux automates sur le même alphabet. Un isomorphisme de* $\mathcal{A}_1$ *vers* $\mathcal{A}_2$ *est une application bijective* φ *de* Q_1 *dans* Q_2 *telle que* $\varphi(I_1) = I_2$, $\varphi(F_1) = F_2$ *et pour tout* $(p_1, a, q_1) \in Q_1 \times \Sigma \times Q_1$, *on* $(p_1, a, q_1) \in \Delta_1$ *si et seulement si* $(\varphi(p_1), a, \varphi(q_1)) \in \Delta_2$.

Ainsi deux automates isomorphes ont le même nombre d'états et sont égaux au renommage près de leurs états.

On montre facilement, avec les notations de la définition 4, que si φ est un isomorphisme de $\mathcal{A}_1$ vers $\mathcal{A}_2$ alors φ^{-1} est un isomorphisme $\mathcal{A}_2$ vers $\mathcal{A}_1$. On peut donc dire, dans ce cas, que les deux automates sont *isomorphes*. La relation d'isomorphisme est une *relation d'équivalence*.

Les automates de la figure 1.4 donnent un exemple d'automates isomorphes, avec $\varphi(1) = 2$, $\varphi(2) = 3$, $\varphi(3) = 4$ et $\varphi(4) = 1$. Les flèches en pointillées répresentent φ.



$$\varphi(\mathcal{A}_1) = \mathcal{A}_2,\ \varphi(1) = 2,\ \varphi(2) = 3,\ \varphi(3) = 4,\ \varphi(4) = 1.$$

FIGURE 1.4 – Deux automates isomorphes

Définition 5 (Automorphisme)**.** *Un automorphisme d'un automate* $\mathcal{A}$ *est un isomorphisme de* $\mathcal{A}$ *dans lui même.*

Pour $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, l'ensemble des automorphismes de $\mathcal{A}$ est un groupe fini noté dans la suite $\mathrm{Aut}(\mathcal{A})$. Pour toute partie Q' de Q, $\mathrm{Aut}_{Q'}(\mathcal{A})$ désigne le sous-ensemble de $\mathrm{Aut}(\mathcal{A})$ constitué des automorphismes φ pour lesquels les éléments de Q' sont des points fixes :

$$\mathrm{Aut}_{Q'}(\mathcal{A}) \stackrel{\text{def}}{=} \{\varphi \in \mathrm{Aut}(\mathcal{A}) \mid \forall q \in Q',\ \varphi(q) = q\}\,.$$

En particulier $\mathrm{Aut}_{\emptyset}(\mathcal{A}) = \mathrm{Aut}(\mathcal{A})$ et $\mathrm{Aut}_{Q}(\mathcal{A})$ se réduit à l'identité de Q.

Dans tous les cas, $\mathrm{Aut}_{Q'}(\mathcal{A})$ est un sous-groupe de $\mathrm{Aut}(\mathcal{A})$.

Par exemple, le groupe d'automorphismes de l'automate représenté par la figure 1.5 admet deux éléments qui sont l'identité de $[\![1, 3]\!]$ et la transposition qui échange 1 et 2.

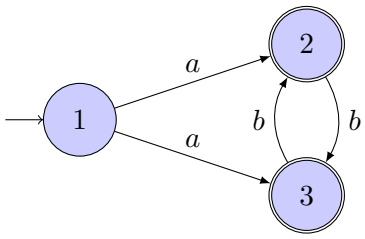


FIGURE 1.5 – Automate $\mathcal{A}$ dont le groupe d'automorphismes, $\mathrm{Aut}(\mathcal{A})$, compte 2 éléments

Le groupe d'automorphismes d'un automate non-déterministe à n états est un sous-groupe du groupe de permutations de $[\![1, n]\!]$, son cardinal est donc un nombre entier naturel diviseur de $n!$.

1.1.4 Minimisation

Il y a une infinité d'automates pour reconnaître un langage rationnel donné. On peut naturellement considérer, parmi l'ensemble des automates déterministes reconnaissant un langage rationnel donné, le sous-ensemble des automates ayant un nombre minimal d'états. On peut démontrer que dans ce sous-ensemble, il y a une unique classe d'isomorphisme : tous les automates déterministes ayant un nombre minimal d'états et reconnaissant un langage rationnel donné sont égaux aux noms des états près.

En revanche, si l'on ne se restreint pas aux automates déterministes, cette unicité à isomorphisme près des automates ayant un nombre minimal d'états est en général fausse : plusieurs automates non-isomorphes de tailles minimales peuvent reconnaître le même langage.

1.1.5 Un résultat sur la combinatoire des automates

Nous utiliserons dans la section 4.2.3, un résultat classique à propos de la combinatoire des automates complets déterministes à n états définis sur un alphabet de k lettres [Kor78].

En notant $\mathfrak{A}_{n,k}$ l'ensemble des automates complets déterministes à n états sur un alphabet de k lettres, on obtient le résultats du théorème 6.

Théorème 6. *Il existe un nombre réel positif γ_k, ne dépendant que de k, tel que :*

$$|\mathfrak{A}_{n,k}| \sim n \cdot \gamma_k \cdot \left\{ {kn \atop n} \right\}.$$

où $\left\{ {m \atop p} \right\}$, dénote le nombre de Stirling de deuxième espèce, c'est-à-dire le nombre de partitions en p classes d'un ensemble à m éléments.

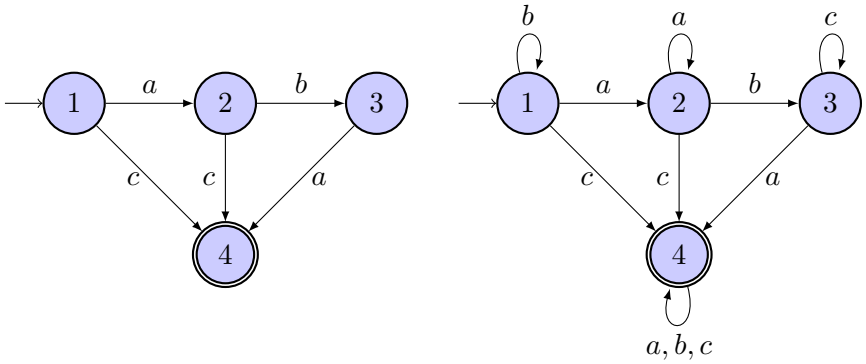
1.2 Automates partiellement ordonnés

Dans cette section on présente les automates partiellement ordonnés de façon formelle. On propose également sur les propriétés de la classe des langages reconnus par ces automates.

Définition 7 (Automate partiellement ordonné)**.** *Un automate partiellement ordonné est un automate fini (Q, Σ, E, I, F) pour lequel il existe une relation d'ordre $\preceq$ sur l'ensemble Q des états telle que si $(p, a, q) \in E$, alors $p \preceq q$.*

Pour un automate partiellement ordonné accessible, l'ensemble I des états initiaux est exactement l'ensemble des états minimaux pour la relation d'ordre $\preceq$.

On peut également noter que si un automate acyclique est partiellement ordonné les deux familles d'automates restent néanmoins distinctes.
La figure 1.6 illustre le propos.



automate acyclique automate partiellement ordonné

FIGURE 1.6 – Un automate acyclique, un automate partiellement ordonné

On peut noter qu'un automate partiellement ordonné "débarrassé" de ses boucles de longueur 1 est un automate acyclique.

Le résultat de la proposition 8 est facile à démontrer grace aux techniques usuelles de construction des automates.

Proposition 8 (automates partiellement ordonnés)**.** *La classe des langages acceptés par les automates partiellement ordonnés est stable par union et intersection.*
Par ailleurs, si on se limite aux automates partiellement ordonnés ***déterministes****, la classe des langages reconnus est stable par union, intersection et passage au complémentaire.*

1.3 Automates à pile

1.3.1 Introduction et intérêt en quelques mots

Les automates à pile sont des outils importants de la théorie des langages, ils reconnaissent les langages hors-contextes définis par les grammaires du même nom. On a coutume de dire que les automates à pile ont des applications nombreuses dans l'étude des langages naturels, l'analyse syntaxique des langages de programmation, l'analyse des programmes. On peut retenir aussi simplement que toute fonction récursive codée dans un langage informatique peut être modélisée par des règles de grammaire adaptées. L'exécution du programme correspondant conduit à la gestion d'une *pile d'appels récursifs*.

On peut illustrer cette idée par le calcul classique du nombre $\pi(p,q)$ de partitions d'un nombre entier naturel p en au plus q parties. Ce calcul peut se présenter sous cette forme :

$$\begin{aligned} \forall p,q \in \mathbb{N}\backslash\{0\}, \quad & (p \geqslant q) \Rightarrow (\pi(p,q) = \pi(p-q,q) + \pi(p,q-1)) \,. \\ \forall p,q \in \mathbb{N}, \quad & (p < q) \Rightarrow (\pi(p,q) = \pi(p,p)) \,. \\ \forall p \in \mathbb{N}, \quad & \pi(p+1,0) = 0. \\ \forall q \in \mathbb{N}, \quad & \pi(0,q) = 1. \end{aligned}$$

Cette présentation formelle du code conduisant au résultat peut aussi être vue comme une grammaire $\mathcal{G}$ pour laquelle les non terminaux sont les symboles $\pi(p,q)$, l'ensemble des terminaux est $\{0,1,+\}$, l'ensemble des règles est constitué de :

$$\begin{aligned} (p \geqslant q > 0) \quad & \pi(p,q) \to \pi(p-q,q) + \pi(p,q-1). \\ (p < q) \quad & \pi(p,q) \to \pi(p,p). \\ (p \in \mathbb{N}) \quad & \pi(p+1,0) \to 0. \\ (q \in \mathbb{N}) \quad & \pi(0,q) \to 1. \end{aligned}$$

On choisit $\pi(p,q)$ comme symbole initial pour obtenir le résultat souhaité sous forme d'une somme de 0 et de 1. Par exemple :

$$\begin{aligned} \pi(3,2) \to \pi(1,2) + \pi(3,1) \to \pi(1,1) + \pi(2,1) + \pi(3,0) \to \pi(0,1) + \pi(1,1) + \pi(2,0) + 0 \\ \to 1 + \pi(0,1) + 0 + 0 \to 1 + 1 + 0 + 0. \end{aligned}$$

1.3.2 Aspects théoriques fondamentaux

Un automate à pile est une machine à états finis qui lit des mots d'un alphabet d'entrée. Cette machine est munie d'une pile de type LIFO qui fonctionne comme une mémoire externe pouvant stocker une quantité non bornée d'information.

Définition 9 (automate à pile)**.** *Un automate à pile est un 7-uplet $\mathcal{A}$ défini par $\mathcal{A} \stackrel{def}{=} (Q, \Sigma \cup \{\varepsilon\}, \Gamma, Z_{\text{init}}, \delta, q_{\text{init}}, F)$ où :*

- *Q est un ensemble fini (l'ensemble des états),*
- *Σ et Γ sont deux alphabets disjoints, Σ est l'alphabet des symboles des mots d'entrée, ε désigne le mot vide, Γ est l'alphabet de pile,*
- *Z_{init} est un symbole appelé symbole initial de la pile,*
- *δ est une relation 5-aire entre les ensembles Q, $\Sigma \cup \{\varepsilon\}$, Γ, Q, Γ^*,*
- *$q_{init} \in Q$ désigne l'état initial,*
- *F est un sous-ensemble, éventuellement vide, de Q ; c'est l'ensemble des états finals.*

Dans la suite de l'exposé, ε désignera le mot vide *en général*, c'est-à-dire aussi bien le mot vide de Γ^* que le mot vide de Σ^*.

Configurations

Sur le plan informel, une *configuration* d'un automate à pile $\mathcal{A}p$ est une description de $\mathcal{A}p$ qui permet de définir à partir de la relation δ les évolutions possibles de l'automate. Sur le plan formel les définitions de cette notion varient un peu suivant les auteurs.

Le premier point de vue consiste à définir une *configuration* par le reste du mot d'entrée restant à lire, l'état courant de l'automate et le contenu de la pile. La configuration *initiale* de $\mathcal{A}p$ est donc un triplet de la forme $(\omega, q_{\text{init}}, Z_{\text{init}})$ où $\omega \in \Sigma^*$ est le mot d'entrée. L'ensemble des configurations de $\mathcal{A}$ est donc le produit cartésien $\Sigma^* \times Q \times \Gamma^*$. Les évolutions possibles de $\mathcal{A}p$ sont alors entièrement décrites par une configuration.

Le second point de vue considère que le mot d'entrée et le reste du mot d'entrée à lire sont des éléments "externes" à $\mathcal{A}p$, une *configuration* de $\mathcal{A}p$ est alors définie comme un

couple de la forme (q, μ) élément de (Q, Γ^*) qui décrit l'état courant de l'automate et le contenu de la pile. Avec ce point de vue la configuration *initiale* de $\mathcal{A}p$ est $(q_{\text{init}}, Z_{\text{init}})$ et l'ensemble des configurations est $Q \times \Gamma^*$. Les évolutions possibles de l'automate à pile ne sont pas alors entièrement décrites par une configuration, elles dépendent de la première lettre qui reste à lire dans le mot d'entrée.

Dans la suite de l'exposé et en particulier dans la définition 10 nous adopterons ce deuxième point de vue. Par ailleurs, l'état de la pile sera décrit par un mot sur l'alphabet Γ dont la dernière lettre représentera le sommet de la pile.

Transitions

La relation δ est un sous-ensemble du produit cartésien $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$, les éléments de δ sont appelés des *transitions*. Une transition est donc un élément de la forme (p, a, X, q, μ).

Si $\tau = (p, a, X, q, \mu)$ est une transition d'un automate à pile $\mathcal{A}p$, on dit que la longueur $|\mu|$ du mot μ qui prend la place de la lettre X sur la pile, est la *taille de sortie* $t(\tau)$ de la transition τ.

On définit alors la *taille de sortie* $t_A(\mathcal{A}p)$ de $\mathcal{A}p$ comme la somme de des tailles de sortie de toutes ses transitions :

$$t_A(\mathcal{A}p) = \sum_{\tau \in \delta} t(\tau) = \sum_{(p,a,X,q,\mu) \in \delta} |\mu|.$$

Lorsque $a = \varepsilon$, on parle d'ε-transition.
Lorsque $\mu = \varepsilon$, on parle de transition de dépilement ou de *pop-transition*.

On dit qu'un automate à pile est à *temps réel* lorsque δ ne contient aucune ε-transition. Sur le plan formel, cela revient à poser $\mathcal{A}p = (Q, \Sigma, \Gamma, Z_{\text{init}}, \delta, q_{\text{init}}, F)$. Il est important de noter qu'avec un mot d'entrée fini, un automate à temps réel ne peut pas évoluer en une suite infinie de configurations. À partir de la configuration initiale, si le mot d'entrée est de longueur n, le calcul terminera en enchaînant ensuite au plus n configurations.

On distingue souvent les pré-conditions (p, a, X) des post-conditions (q, μ) dans les transitions de $\mathcal{A}$ pour traduire l'idée que lorsque l'automate est dans l'état p, que la prochaine lettre du mot d'entrée à lire est a, que le sommet de pile est X, alors l'automate est susceptible de passer dans l'état q en dépilant X pour empiler le mot μ de Γ^* sur la pile. On peut alors considérer δ comme une relation binaire entre les ensembles $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ et $Q \times \Gamma^*$ ou comme une fonction de $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ sur l'ensemble des parties finies de $Q \times \Gamma^*$. On pourra donc ainsi donner du sens à la notation $(q, \mu) \in \delta(p, a, X)$.

On dit qu'un automate à pile $\mathcal{A}p$ est *complet* lorsque le domaine de la relation δ, vue comme relation binaire entre les ensembles $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ et $Q \times \Gamma^*$ est égale à $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$. Pour un automate à pile à temps réel $\mathcal{A}p$, on dit que $\mathcal{A}p$ est complet lorsque le domaine de δ est égal à $Q \times \Sigma \times \Gamma$. En clair, quelle que soit la configuration de $\mathcal{A}p$ considérée, quelle que soit la première lettre a restant à lire sur le mot d'entrée, il existe une évolution de $\mathcal{A}p$.

On dit qu'un automate à pile temps réel $\mathcal{A}p$ est *déterministe*, lorsque la relation δ, vue comme relation binaire entre les ensembles $Q \times \Sigma \times \Gamma$ et $Q \times \Gamma^*$ est fonctionnelle. On note alors $(q, \mu) = \delta(p, a, X)$ à la place de $(q, \mu) \in \delta(p, a, X)$. Dans une configuration donnée, pour une première lettre $a \in \Sigma$ restant à lire dans le mot d'entrée, il n'y a qu'une seule évolution possible de $\mathcal{A}p$. L'automate à pile temps réel $\mathcal{A}p$ est complet et déterministe lorsque δ peut être considérée comme une *fonction totale* de $Q \times \Sigma \times \Gamma$ sur $Q \times \Gamma^*$.

Dans le cas d'un automate à pile déterministe à temps réel, on donne la définition suivante qui traduit l'enchaînement des configurations à partir de la configuration initiale et un mot d'entrée.

Définition 10 (Configurations consécutives)**.** *On considère $\mathcal{A}p$ un automate à pile* ***déterministe et à temps réel****, $a \in \Sigma$. On dit que deux configurations (q_1, μ_1) et (q_2, μ_2) de $\mathcal{A}p$ sont a-consécutives si les conditions suivantes sont vérifiées :*

- *$\mu_1 \neq \varepsilon$. On peut donc considérer $\mu_3 \in \Gamma^*$ et $X \in \Gamma$ tels que $\mu_1 = \mu_3 X$*
- *Il existe $\mu_4 \in \Gamma^*$ tel que $\mu_2 = \mu_3\mu_4$ et $(q_2, \mu_4) \in \delta(q_1, a, X)$.*

Si c'est le cas, on note $(q_1, \mu_1) \models_a (q_2, \mu_2)$.

Deux configurations (q_1, μ_1) et (q_2, μ_2) sont dites consécutives s'il existe $a \in \Sigma$ tel que $(q_1, \mu_1) \models_a (q_2, \mu_2)$.

Si c'est le cas, on note simplement $(q_1, \mu_1) \models (q_2, \mu_2)$.

1.3.3 Automate sous-jacent et notions théoriques associées

Dans cette section on explique comment les notions d'accessibilité et d'atteignabilité découlent du fait qu'un automate à pile peut être vu comme un automate de mots usuel à condition de faire abstraction des opérations de pile.

Automate sous-jacent

Abstraction faite des opérations de pile d'un automate à pile $\mathcal{A}p = (Q, \Sigma, \Gamma, Z_{\text{init}}, \delta, q_{\text{init}}, F)$, tout calcul réalisé sur $\mathcal{A}p$ revient à un calcul sur un automate fini de mots pour lequel l'alphabet de mot est le produit cartésien $\Sigma \times \Gamma$ des alphabets de $\mathcal{A}p$. Plus formellement, on a la définition suivante :

Définition 11 (Automate sous-jacent)**.** *On considère $\mathcal{A}p = (Q, \Sigma, \Gamma, Z_{\text{init}}, \delta, q_{\text{init}}, F)$ un automate à pile à temps réel, on appelle automate sous-jacent à $\mathcal{A}p$ l'automate de mots $\mathcal{A}p' = (Q, \Sigma \times \Gamma, \delta', q_{\text{init}}, F)$, où δ' est une relation entre les ensembles Q, $\Sigma \times \Gamma$ et Q définie par :*

$$\forall (p, (a, X), q) \in Q \times (\Sigma \times \Gamma) \times Q, \quad ((p, (a, X), q) \in \delta') \Leftrightarrow (\exists \mu \in \Gamma^*, \quad (p, a, X, q, \mu) \in \delta)\,.$$

Avec les notations de la définition 11, si l'automate $\mathcal{A}p$ est déterministe alors l'automate $\mathcal{A}p'$ est déterministe. La réciproque n'est pas vraie en général. En revanche, $\mathcal{A}p$ est complet si et seulement si $\mathcal{A}p'$ est complet.

La figure 1.7 donne un exemple d'un automate à pile $\mathcal{A}p_{ex}$ déterministe à temps réel complet de taille de sortie égale à 11, pour lequel :

- $Q = \{0, 1\}$,
- $\Sigma = \{a, b\}$, $\Gamma = \{Z, X\}$,
- δ est défini par l'ensemble :
  $\{(0, (a, X)) \mapsto (1, Z), (0, (b, X)) \mapsto (0, X), (0, (a, Z)) \mapsto (0, ZZX), (0, (b, Z)) \mapsto (0, ZX),$
  $(1, (a, X)) \mapsto (1, XX), (1, (b, X)) \mapsto (0, \varepsilon), (1, (a, Z)) \mapsto (0, XZX), (1, (b, Z)) \mapsto (1, \varepsilon)\}$.
- $q_{\text{init}} = 0, \quad Z_{\text{init}} = Z, \quad F = \{1\}$.

Son automate sous-jacent est représenté 1.8.

Accessibilité et atteignabilité

La notion d'automate sous-jacent permet de définir l'accessibilité d'un automate à pile à temps réel :

Définition 12 (Automate à pile à temps réel, accessibilité)**.** *On dit qu'un automate à pile à temps réel $\mathcal{A}p$ est accessible si et seulement si son automate sous-jacent $\mathcal{A}p'$ est accessible.*

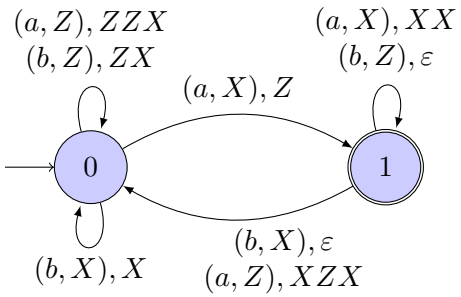
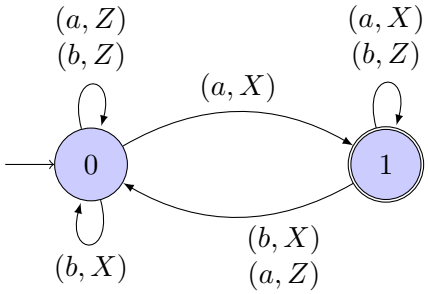


FIGURE 1.7 – $\mathcal{A}_{ex}$, automate à pile déterministe, temps réel, complet



$$
\begin{aligned}
Q &= \{0, 1\}, \\
\text{Alphabet} &= \{a, b\} \times \{Z, X\}, \\
\delta' &= \{(0, (a, X)) \mapsto 1, (0, (b, X)) \mapsto 0, \\
&\quad (0, (a, Z)) \mapsto 0, (0, (b, Z)) \mapsto 0, \\
&\quad (1, (a, X)) \mapsto 1, (1, (b, X)) \mapsto 0, \\
&\quad (1, (a, Z)) \mapsto 0, (1, (b, Z)) \mapsto 1\} \\
&\quad q_{\text{init}} = 0, \quad F = \{1\}.
\end{aligned}
$$

FIGURE 1.8 – Automate sous-jacent à $\mathcal{A}_{ex}$

Avec les notations précédentes, il faut bien noter que s'il existe un chemin entre q_{init} et tout état de Q pour l'automate $\mathcal{A}p'$ ce n'est pas forcément le cas dans l'automate $\mathcal{A}p$ pour lequel les opérations de pile sont contraignantes. Pour exprimer le fait qu'il existe un calcul dans l'automate à pile qui conduit de l'état initial à un état $q \in Q$, on est conduit à définir une autre notion :

Définition 13 (Automate à pile à temps réel, atteignabilité)**.** *On considère $\mathcal{A}p = (Q, \Sigma, \Gamma, Z_{\text{init}}, \delta, q_{\text{init}}, F)$ un automate à pile à temps réel, $q \in Q$ un état. On dit que q est atteignable s'il existe une suite finie $((q_k, \mu_k))_{k \in [\![0,n]\!]}$ de configurations telles que :*

$$(q_0, \mu_0) = (q_{\text{init}}, Z_{\text{init}}), \quad q_n = q, \quad \forall k \in [\![0, n]\!], \quad (q_k, \mu_k) \models (q_{k+1}, \mu_{k+1}).$$

Si c'est le cas et si par ailleurs $\mu_n = \varepsilon$, on dit que q est atteignable par pile vide.

On dit que $\mathcal{A}p$ est atteignable lorsque pour tout état $q \in Q$, q est atteignable.

Le résultat de la proposition 14 découle naturellement des considérations précédentes.

Proposition 14 (accessibilité, atteignabilité)**.** *On considère $\mathcal{AP} = (Q, \Sigma, \Gamma, Z_{\text{init}}, \delta, q_{\text{init}}, F)$ un APDTR et $p \in Q$, alors :*

$$(p \text{ est atteignable par pile vide}) \Rightarrow (p \text{ est atteignable}) \Rightarrow (p \text{ est accessible}).$$

Les réciproques sont fausses dans le cas général.

En considérant l'automate de la figure 1.9, on constate que l'état 3 est accessible, puisqu'il suffit, depuis l'état initial 0, de lire la lettre (a, Z) de l'alphabet produit $\Sigma \times \Gamma$ pour accéder à l'état 3. Si le symbole initial de pile est Z, le seul mouvement possible de l'automate est réalisé lorsque la lettre a est lue. Si c'est le cas, l'automate passe de l'état 0 à l'état 3.

En revanche, si le symbole initial de pile est X, la transition de l'état initial 0 à l'état 3 ne sera jamais empruntée.

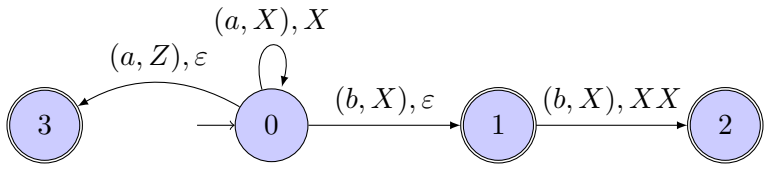


FIGURE 1.9 – État accessible, atteignable, différentes conditions d'acceptation

Par ailleurs le langage accepté (voir section suivante) par pile vide, ou par pile vide et état final avec comme symbole de fond de pile X est $\{b\}$, alors que le langage accepté par état final est le langage régulier $a^*(b + bb)$.

Dans ce qui suit, sauf précision, tous les automates à pile considérés sont à temps réel.

Conditions d'acceptation, langages reconnus

Pour un mot $\omega = a_1 \dots a_k \in \Gamma^*$ et pour un automate à pile $\mathcal{A}p$, on distingue trois principaux modes d'acceptation :

- Par pile vide : ω est accepté s'il existe une suite $c_0, \dots, c_k$ de configurations vérifiant $c_0 = (q_{\text{init}}, Z_{\text{init}})$, c_k de la forme (q, ε) où $q \in Q$ et :

$$\forall i \in [\![1, k]\!], \quad c_{i-1} \models_{a_i} c_i.$$

- Par état final : ω est accepté s'il existe une suite $c_0, \dots, c_k$ de configurations vérifiant $c_0 = (q_{\text{init}}, Z_{\text{init}})$, c_k de la forme (q, μ) où $(q, \mu) \in F \times \Gamma^*$ et :

$$\forall i \in [\![1, k]\!], \quad c_{i-1} \models_{a_i} c_i.$$

- Par état final et pile vide : ω est accepté s'il existe une suite $c_0, \dots, c_k$ de configurations vérifiant $c_0 = (q_{\text{init}}, Z_{\text{init}})$, c_k de la forme (q, ε) où $q \in F$ et :

$$\forall i \in [\![1, k]\!], \quad c_{i-1} \models_{a_i} c_i.$$

Par exemple, pour le APDTR décrit à la figure 1.9, on a :

$$(0, ZX) \models_a (0, ZX), \quad (0, XZ) \models_a (3, X), \quad (1, ZX) \models_b (2, ZXX).$$

Le langage reconnu par $\mathcal{A}p$ par pile vide, par état final ou état final et pile vide respectivement, est l'ensemble des mots de Σ^* acceptés par pile vide, état final, pile vide et état final respectivement. Dans le cas de reconnaissance par pile vide, le langage reconnu ne dépend pas de la définition de l'ensemble F.

Il est légitime de se poser la question de l'influence de la condition d'acceptation sur la famille des langages reconnus par les automates à pile. Dans le cas des automates à pile en général (déterministes ou non, à temps réel ou non), le résultat important est que la condition d'acceptation n'est pas déterminante. En notant $L_v(\mathcal{A}p)$, $L_f(\mathcal{A}p)$ et $L_{vf}(\mathcal{A}p)$ les langages reconnus par pile vide, état final, pile vide et état final respectivement, pour tous $\alpha, \beta \in \{v, f, vf\}$, pour tout automate à pile $\mathcal{A}p$, il existe un automate à pile $\mathcal{A}p^\circ$ tel que $L_\alpha(\mathcal{A}p) = L_\beta(\mathcal{A}p^\circ)$. D'une certaine façon, les conditions d'acceptation sont donc équivalentes dans le sens où elles n'influent pas sur l'ensemble des langages reconnus. Il est bien connu que la famille des langages reconnus est celle des langages hors-contextes.

La situation est néanmoins plus délicate avec les automates à pile déterministes. Même si ce n'est pas le cœur de ce travail, on peut retenir que dans le cas des automates déterministes les langages reconnus par pile vide sont nécessairement *prefix-free*. Par ailleurs, la famille des langages reconnus par les automates à pile déterministes, quel que soit le mode d'acceptation, est une sous-famille propre de la famille des langages hors-contextes.

Les automates des figures 1.10 et 1.11 proposent deux exemples d'automates acceptant des langages hors-contextes non-réguliers.

Pour le premier automate (1.10), le langage reconnu par pile vide est le langage L_1 défini par $L_1 = \{a^n b^{2n},\ n \geqslant 0\}$.

Le langage reconnu par état final est le langage L_2 défini par $L_2 = \{a^n b^p,\ 0 \leqslant p \leqslant 2n\}$. En particulier, dans les deux cas, le mot vide est reconnu. Ces deux langages ne sont pas réguliers.

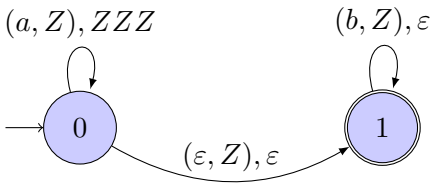


FIGURE 1.10 – Langage hors-contexte avec ε-transition

Le deuxième automate (1.11) reconnaît par pile vide le langage $L_1 \backslash \{\varepsilon\}$ et par état final le langage $L_2 \backslash \{\varepsilon\}$.

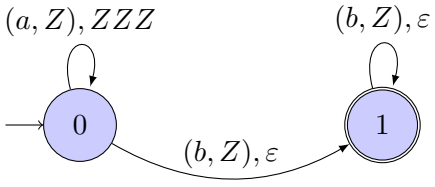


FIGURE 1.11 – Langage hors-contexte sans ε-transition

Isomorphisme d'automates à pile

Dans ce travail il s'agira dans la partie *contributions* d'engendrer des automates à pile *à isomorphisme près*.

Deux automates à pile généraux $\mathcal{A}p_1$ et $\mathcal{A}p_2$ définis par $\mathcal{A}p_1 = (Q_1, \Sigma \cup \{\varepsilon\}, \Gamma, Z_{\text{init}}, \delta_1, q_{\text{init1}}, F_1)$ et $\mathcal{A}p_2 = (Q_2, \Sigma \cup \{\varepsilon\}, \Gamma, Z_{\text{init}}, \delta_2, q_{\text{init2}}, F_2)$ sont isomorphes si et seulement si il existe une bijection $\varphi : Q_1 \to Q_2$ telle que :

- $\varphi\left(q_{\text{init1}}\right) = q_{\text{init2}}$
- $\varphi\left(F_1\right) = F_2$
- Pour tout $(p, a, X) \in Q_1 \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$, pour tout $(q, \mu) \in Q_1 \times \Gamma^*$, on a :

$$\left((q, \mu) \in \delta_1\left(p, a, X\right)\right) \Leftrightarrow \left((\varphi(q), \mu) \in \delta_2\left(\varphi(p), a, X\right)\right).$$

Dans le cas des automates déterministes les relations d'appartenance sont remplacées par des égalités.

Chapitre 2

Analyse combinatoire, génération aléatoire

Sommaire

2.1 La méthode symbolique

La méthode symbolique ainsi qu'une foule de prolongements est exposée en détails dans [FS08] qui est un ouvrage de référence. Le principe de la méthode symbolique consiste à décrire une classe combinatoire par sa série génératrice (souvent définie récursivement), afin d'automatiser le dénombrement des objets de taille fixée de cette classe.

De façon schématique, la méthode symbolique constitue un point d'entrée à la fois pour la méthode récursive et pour la méthode de Boltzmann. La première s'appuie en effet sur la valeur des coefficients de la série génératrice associée à la classe combinatoire considérée. La mise au point d'un générateur de Boltzmann, passe quant à elle, par les valeurs de la séries génératrice en certains points.

2.1.1 Le concept de base de la méthode symbolique

Une classe combinatoire est un ensemble $\mathcal{A}$, au plus dénombrable, muni d'une application f (la taille) qui vérifie :

- Pour tout $a \in \mathcal{A}$, $\quad f(a) \in \mathbb{N}$.
- Pour tout $n \in \mathbb{N}$, l'ensemble $f^{-1}(\{n\})$ des objets de taille n est fini.

À toute classe combinatoire $\mathcal{A}$ on associe sa *série génératrice* (ordinaire) $A(z)$ définie par :

$$\boxed{A(z) = \sum_{n=0}^{\infty} a_n z^n, \quad a_n = \left|f^{-1}(\{n\})\right|.}$$

Le nombre entier a_n est donc le cardinal de l'ensemble des objets de taille n dans la classe combinatoire $\mathcal{A}$.

On peut donc encore écrire de façon équivalente :

$$\boxed{A(z) = \sum_{\alpha \in \mathcal{A}} z^{|\alpha|}.}$$

On dit encore que la variable z **marque la taille** dans la série génératrice.

On note par ailleurs :

$$\boxed{[z^n]\left(\sum_{n\geq 0} a_n z^n\right) := a_n.}$$

2.1.2 Quelques classes combinatoires et leurs séries génératrices

- On note $\mathcal{W}$ l'ensemble des mots binaires, où la taille des mots est définie par leur longueur, on a :

$$\boxed{W(z) = \sum_{n\geq 0} 2^n z^n = \frac{1}{1-2z}.}$$

- On note $\mathcal{P}$ l'ensemble des permutations, pour tout $\sigma \in \mathfrak{S}_n$, la taille de σ est égale à n, on a :

$$\boxed{P(z) = \sum_{n\geq 0} n! z^n.}$$

- On note $\mathcal{T}$ l'ensemble des triangulations des polygones réguliers, la taille est donnée par le nombre de cotés du polygone :

$$\boxed{T(z) = \sum_{n\geq 0} \frac{1}{n+1}\binom{2n}{n} z^n = \frac{1-\sqrt{1-4z}}{2z}.}$$

La figure 2.1 illustre cette classe en en représentant quelques objets.

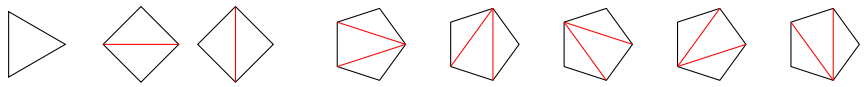


FIGURE 2.1 – Triangulations, exemples des triangles, des carrés et des pentagones

2.1.3 Le cas des structures étiquetées

Beaucoup de classes combinatoires se présentent naturellement comme des structures étiquetées. Pour un graphe planaire enraciné de n sommets, on peut distinguer les sommets en les étiquetant par les entiers naturels de 1 à n. On est alors conduit à distinguer jusqu'à $n!$ graphes (lorsque le groupe d'automorphismes est trivial) là où dans le modèle non-étiqueté on en distinguait qu'un seul. On est conduit naturellement à introduire les séries génératrices exponentielles associées à ces structures. Pour une suite (a_n), la série génératrice exponentielle $A(z)$ associée est définie par :

$$\boxed{A(z) = \sum_{n\geq 0} a_n \frac{z^n}{n!}.}$$

En particulier :

$$\boxed{a_n = n! \cdot [z^n]A(z).}$$

ce qui est conforme à l'introduction informelle faite à propos de ce modèle.

Les exemples fondamentaux sont les suivants :

- La classe $\mathcal{P}$ des permutations :

$$\mathcal{P} = \left\{ \varepsilon, \begin{array}{|cc|}\hline 1 & 2 \\ \hline 2 & 1 \\ \hline \end{array}, \begin{array}{|ccc|}\hline 1 & 2 & 3 \\ \hline 1 & 3 & 2 \\ \hline 2 & 1 & 3 \\ \hline 2 & 3 & 1 \\ \hline 3 & 1 & 2 \\ \hline 3 & 2 & 1 \\ \hline \end{array} \dots \right\}.$$

  La série génératrice exponentielle associée est :

$$\boxed{P(z) = \sum_{n\geq 0} n! \frac{z^n}{n!} = \sum_{n\geq 0} z^n = \frac{1}{1-z}.}$$

- La classe combinatoire $\mathcal{U}$ des urnes. Les urnes de taille n peuvent être considérées comme les "boites" contenant n boules distinctes (distinguables) numérotées de 1 à n :

$$\mathcal{U} = \left\{ \varepsilon, \boxed{\textcircled{1}}, \boxed{\textcircled{1}\ \textcircled{2}}, \boxed{\begin{array}{ll}\textcircled{1} & \textcircled{2} \\ \textcircled{3} & \end{array}}, \dots \right\}.$$

  La série génératrice exponentielle associée est :

$$\boxed{U(z) = \sum_{n\geq 0} 1 \frac{z^n}{n!} = \exp(z).}$$

- La classe combinatoire $\mathcal{C}$ des graphes circulaires orientés :

$$\mathcal{C} = \left\{ \textcircled{1}, \begin{array}{c}\textcircled{2} \\ \textcircled{1}\end{array}, \begin{array}{c}\textcircled{2} \\ \textcircled{1} \\ \textcircled{3}\end{array}, \begin{array}{c}\textcircled{2} \\ \textcircled{1} \\ \textcircled{3}\end{array}, \dots \right\}.$$

  Puisqu'un cycle est défini par la succession des entiers qui suivent 1 on a $C_n = (n-1)!$ et la série génératrice exponentielle associée est :

$$\boxed{C(z) = \sum_{n\geq 1} (n-1)! \frac{z^n}{n!} = \sum_{n\geq 1} \frac{z^n}{n} = \ln\left(\frac{1}{1-z}\right).}$$

2.1.4 De la structure à la série génératrice

Des classes combinatoires élémentaires peuvent servir à construire d'autres classes combinatoires par union (disjointe), produit cartésien, suite, parties d'un ensemble, multi-ensemble, cycle. On dit que ces constructions sont *admissibles*. Les deux modèles, ordinaires et exponentiels peuvent être utilisés en fonction des contextes.

Le tableau de la figure 2.2 (dans lequel φ désigne l'indicatrice d'Euler) résume les résultats qui permettent de passer de la spécification de la structure combinatoire à la série génératrice associée.

classe combinatoire	Série génératrice ordinaire	Série génératrice exponentielle
$\mathcal{A} = \mathcal{B} + \mathcal{C}$	$A(z) = B(z) + C(z)$	$\hat{A}(z) = \hat{B}(z) + \hat{C}(z)$
$\mathcal{A} = \mathcal{B} \times \mathcal{C}$	$A(z) = B(z) \cdot C(z)$	$\hat{A}(z) = \hat{B}(z) \cdot \hat{C}(z)$
$\mathcal{A} = SEQ(\mathcal{B})$	$A(z) = \dfrac{1}{1 - B(z)}$	$\hat{A}(z) = \dfrac{1}{1 - \hat{B}(z)}$
$\mathcal{A} = PSET(\mathcal{B})$	$A(z) = \begin{cases} \prod_{n\geq 1}(1+z^n)^{B_n} \\ \exp\left(\sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k} B\left(z^k\right)\right) \end{cases}$	$\hat{A}(z) = \exp\left(\hat{B}(z)\right)$
$\mathcal{A} = MSET(\mathcal{B})$	$A(z) = \begin{cases} \prod_{n\geq 1}(1-z^n)^{-B_n} \\ \exp\left(\sum_{k=1}^{\infty} \frac{1}{k} B\left(z^k\right)\right) \end{cases}$	–
$\mathcal{A} = CYC(\mathcal{B})$	$A(z) = \sum_{k=1}^{\infty} \frac{\varphi(k)}{k} \ln \frac{1}{1 - B\left(z^k\right)}$	$\hat{A}(z) = \ln\left(\dfrac{1}{1 - \hat{B}(z)}\right)$

FIGURE 2.2 – De la structure à la série génératrice

2.1.5 Quelques exemples simples

- On considère la classe atomique constituée d'un seul élément : $\mathcal{Z} := \{\bullet\}$. Alors $\mathcal{A} = \mathcal{Z} + \mathcal{Z} \times \mathcal{Z}$ est l'ensemble constitué des deux éléments $\bullet$ et $(\bullet, \bullet)$ représenté par $\bullet\!-\!\bullet$. La classe combinatoire $\mathcal{C} = SEQ(\mathcal{A})$ contient :

  $\bullet$, $\bullet\,\bullet$, $\bullet\!-\!\bullet$, $\bullet\,\bullet\!-\!\bullet$, $\bullet\,\bullet\,\bullet$, $\bullet\!-\!\bullet\,\bullet$, $\bullet\,\bullet\!-\!\bullet\,\bullet$, $\bullet\!-\!\bullet\!-\!\bullet\!-\!\bullet$, $\bullet\!-\!\bullet\,\bullet\,\bullet$, $\bullet\,\bullet\,\bullet\!-\!\bullet$, $\bullet\,\bullet\,\bullet\,\bullet$, ...

  Les éléments de taille n de $\mathcal{C}$ peuvent s'interpréter comme les recouvrements de l'intervalle d'entiers $[0, n]$ par des intervalles de longueur 1 ou 2.

  On a $A(z) = z + z^2$ et :

  $$\boxed{C(z) = \frac{1}{1 - A(z)} = \frac{1}{1 - z - z^2}.}$$

  La série $C(z)$ est celle des nombres de Fibonacci.
- Les arbres enracinés peuvent être considérés comme les objets d'une classe combinatoire $\mathcal{G}$ dont la taille est définie par le nombre de nœuds. Un arbre étant défini

par son nœud racine et la forêt qui lui est associée, on a :

$$\boxed{\mathcal{G} = \mathcal{Z} \times SEQ(\mathcal{G}).}$$

On a donc :

$$G(z) = \frac{z}{1 - G(z)}.$$

On obtient, en ne conservant que la solution conduisant à des coefficients positifs :

$$\boxed{G(z) = \frac{1}{2}\left(1 - \sqrt{1-4z}\right) = \sum_{n\geq 1} \frac{1}{n}\binom{2n-2}{n-1} z^n.}$$

On notera que :

$$\boxed{G_n = T_{n-1}.}$$

- En notant $\mathcal{I}$ la classe combinatoire des entiers naturels non-nuls, la taille d'un objet étant égale à sa valeur, on a donc :

$$\boxed{I(z) = \sum_{n\geq 1} z^n = \frac{z}{1-z}.}$$

Pour tout entier naturel non-nul n, on appelle **composition de** n tout k-uplet $(x_1, \ldots, x_k)$ d'entiers naturels non-nuls tel que :

$$n = \sum_{i=1}^{k} x_i.$$

Pour tout entier naturel non-nul n, on appelle **partition de** n tout k-uplet $(x_1, \ldots, x_k)$ d'entiers naturels non-nuls tel que :

$$n = \sum_{i=1}^{k} x_i \quad \wedge \quad x_1 \geq x_2 \geq \cdots \geq x_k \geq 1.$$

En notant $\mathcal{CP}$ la classe combinatoire des **compositions**, on a :

$$\boxed{\mathcal{CP} = SEQ(\mathcal{I}).}$$

On a donc directement :

$$\boxed{CP(z) = \frac{1}{1 - \frac{z}{1-z}} = (1-z)\cdot\frac{1}{1-2z} = \sum_{n\geq 0} 2^n z^n - \sum_{n\geq 0} 2^n z^{n+1}.}$$

On retrouve le résultat combinatoire connu :

$$\boxed{\forall n \in \mathbb{N}\backslash\{0\}, \quad CP_n = 2^{n-1}.}$$

En notant $\mathcal{P}$ la classe combinatoire des **partitions**, on a :

$$\boxed{\mathcal{P} = MSET(\mathcal{I}).}$$

On a donc directement :

$$\boxed{P(z) = \prod_{m=1}^{\infty} \frac{1}{1-z^m} = 1 + z + 2z^2 + 3z^3 + 5z^4 + 7z^5 + 11z^6 + 15z^7 + 22z^8 + \ldots}$$

Il n'y a pas de formule explicite de P_n, on peut retenir $P_n = \mathrm{e}^{O(\sqrt{n})}$.

- En notant $\mathcal{P}_e$ la classe combinatoire des partitions d'ensembles, on a :

$$\boxed{\mathcal{P}_e = PSET\left(PSET_{\geq 1}(\mathcal{Z})\right).}$$

La série génératrice exponentielle $\hat{A}(z)$ associée à $PSET_{\geq 1}(\mathcal{Z})$ est définie par :

$$\boxed{\hat{A}(z) = \exp(z) - 1.}$$

On a donc immédiatement :

$$\boxed{\hat{P}_e(z) = \exp(\exp(z) - 1) = 1 + z + z^2 + \frac{5}{6}z^3 + \frac{5}{8}z^4 + \frac{13}{30}z^5 + \frac{203}{720}z^6 + \frac{877}{5040}z^7 + \frac{23}{224}z^8 + \frac{1007}{17280}z^9 + \dots}$$

2.2 Génération aléatoire par la méthode récursive

Pour un exposé clair et détaillé (en français) de la méthode récursive, le lecteur pourra se rapporter par exemple à [Piv08] et [Dav10]. Nous nous contentons ici de rappels synthétiques largement inspirés de ces références et illustrés par le premier exemple de la section 2.1.5.

2.2.1 Principe général de la méthode récursive

La méthode récursive consiste à piloter la génération aléatoire d'objets d'une classe combinatoire donnée en s'appuyant d'abord sur une spécification récursive standard de cette classe. Cette description permet un passage presque direct de la spécification à l'implémentation du générateur.

Le succès du principe général repose sur le fait qu'une classe combinatoire $\mathcal{C}$ se présente en général comme la composée de sous-structures. La décomposition en sous-structures passe par une spécification standard qui n'utilise que des unions, des produits et un opérateur de pointage.

La navigation parmi les objets de taille n de la structure peut alors être représentée comme un parcours d'arbre. Lorsque la classe combinatoire $\mathcal{C}$ se présente, par exemple, comme une union de deux classes combinatoires $\mathcal{A}$ et $\mathcal{B}$, la première étape du tirage aléatoire d'une structure de taille n consiste à tirer une $\mathcal{A}$ structure de taille n avec une probabilité $|\mathcal{A}_n|/|\mathcal{C}_n|$ et une $\mathcal{B}$ structure de taille n avec la probabilité (complémentaire) $|\mathcal{B}_n|/|\mathcal{C}_n|$. Les arêtes de l'arbre associé à la structure sont donc étiquetées par les probabilités de transition. La figure 2.3 illustre cette idée.

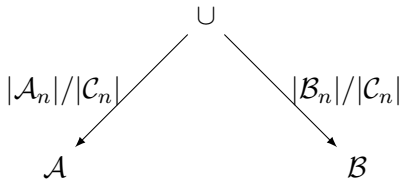


FIGURE 2.3 – Structure combinatoire $\mathcal{C}_n$ union de deux structures

Il ne s'agit évidemment pas de construire de façon exhaustive l'arbre associé à la structure mais de déterminer à chaque nœud les probabilités de transition vers un des nœuds fils jusqu'à parvenir à une feuille de l'arbre qui sera le résultat du tirage aléatoire.

L'algorithme de génération récursive $\texttt{GenRec}(\texttt{C},\texttt{n})$ d'une $\mathcal{C}$-structure de taille n fonctionne alors sous le principe du pseudo-code récursif suivant :

$\texttt{GenRec}(\mathcal{C},\texttt{n})$:

- $\mathcal{C}=\mathcal{E}$:
  $\texttt{Si}\ \ \texttt{n}=0\ \texttt{alors renvoyer}\ \mathcal{E}$
- $\mathcal{C}=\mathcal{Z}$:
  $\texttt{Si}\ \ \texttt{n}=1\ \texttt{alors renvoyer}\ \mathcal{Z}$
- $\mathcal{C}=\mathcal{A}+\mathcal{B}$:
  $\texttt{u}\leftarrow\texttt{random}(0,1)$
  $\texttt{Si}\quad \texttt{u}<\texttt{a}_\texttt{n}/\texttt{c}_\texttt{n}\quad \texttt{alors renvoyer}$
  $\texttt{GenRec}(\mathcal{A},\texttt{n})$
  $\texttt{Sinon renvoyer}\ \ \texttt{GenRec}(\mathcal{B},\texttt{n})$
- $\mathcal{C}=\mathcal{A}\cdot\mathcal{B}$:
  $\texttt{u}\leftarrow\texttt{random}(0,1)$
  $\texttt{k}\leftarrow 0;\texttt{s}\leftarrow(\texttt{a}_0\texttt{b}_\texttt{n})/\texttt{c}_\texttt{n}$
  $\texttt{Tant que}\ \ \texttt{u}>\texttt{s}\ \ \texttt{Faire}$
  $\texttt{k}\leftarrow\texttt{k}+1;\texttt{s}\leftarrow\texttt{s}+(\texttt{a}_\texttt{k}\texttt{b}_{\texttt{n}-\texttt{k}})/\texttt{c}_\texttt{n}$
  $\texttt{Fin Faire}$
  $\texttt{renvoyer}$
  $\langle\texttt{GenRec}(\mathcal{A},\texttt{k}),\texttt{GenRec}(\mathcal{B},\texttt{n}-\texttt{k})\rangle$

Dans le cas où $\mathcal{C}=\mathcal{A}\cdot\mathcal{B}$, on emprunte une des $n+1$ branches issues de la racine. On a :

$$\texttt{c}_\texttt{n}=\sum_{\texttt{k}=0}^{\texttt{n}}\texttt{a}_\texttt{k}\texttt{b}_{\texttt{n}-\texttt{k}}$$

Pour $k\in[\![0,n]\!]$ la probabilité d'emprunter la k-ième branche est $(\texttt{a}_\texttt{k}\texttt{b}_{\texttt{n}-\texttt{k}})/\texttt{c}_\texttt{n}$. En notant s_k le nombre réel de l'intervalle $[0,1]$ défini par $s_{-1}=0$ et :

$$\forall\texttt{k}\in[\![0,\texttt{n}]\!],\quad \texttt{s}_\texttt{k}\overset{\text{def}}{=}\sum_{\texttt{i}=0}^{\texttt{p}}(\texttt{a}_\texttt{i}\texttt{b}_{\texttt{n}-\texttt{i}})/\texttt{c}_\texttt{n}$$

on emprunte la k-ième branche qui vérifie :

$$\texttt{u}\in\,]\texttt{s}_{\texttt{k}-1},\texttt{s}_\texttt{k}]$$

2.2.2 Exemple d'utilisation de la méthode récursive

On propose d'illustrer la méthode récursive par le premier exemple très simple de la section 2.1.5. Dans cette exemple, la structure combinatoire $\mathcal{C}$ vérifie : $\mathcal{C}=SEQ(\mathcal{A})$, où $\mathcal{A}=\mathcal{Z}\times\mathcal{Z}$. La classe combinatoire peut être définie récursivement par :

$$\mathcal{C}=\mathcal{A}\times\mathcal{C}=\mathcal{Z}\times\mathcal{C}+\mathcal{Z}^2\times\mathcal{C}.$$

On en déduit la formule qui permet de retrouver l'idée que la série génératrice associée est celle de Fibonacci :

$$\forall n\in\mathbb{N},\quad c_{n+2}=c_{n+1}+c_n.$$

Pour engendrer des structures de taille n, on commence donc par calculer c_k pour $k\in[\![0,n]\!]$. Le générateur lui même s'implémente en pseudo-code de la façon suivante :

$\texttt{GenRec}(\mathcal{C},\texttt{n})$:
$\texttt{Si}\ \ \texttt{n}=0\ \ \texttt{Alors renvoyer}\ \ \emptyset$
$\texttt{Si}\ \ \texttt{n}=1\ \ \texttt{Alors renvoyer}\ \ \bullet$
$\texttt{Sinon}$
$\texttt{Avec la probabilité}\ \ \texttt{c}_{\texttt{n}-1}/\texttt{c}_\texttt{n}\ \ \texttt{renvoyer}\ \ \bullet,\texttt{GenRec}(\mathcal{C},\texttt{n}-1)$
$\texttt{Avec la probabilité}\ \ \texttt{c}_{\texttt{n}-2}/\texttt{c}_\texttt{n}\ \ \texttt{renvoyer}\ \ \bullet\!\!-\!\!\bullet,\texttt{GenRec}(\mathcal{C},\texttt{n}-2)$

Pour $n=5$ on obtient par exemple l'arbre de la figure 2.4.

Comme déjà signalé, l'arbre de transition n'est évidemment pas construit de façon exhaustive, sans quoi la méthode n'aurait pas vraiment d'intérêt. À chaque nœud interne, l'algorithme permet un tirage aléatoire adéquate du nœud suivant qui est le seul à être explicité. L'algorithme renvoie un résultat une fois parvenu à une feuille (en rouge sur la figure 2.4).

2.2.3 En pratique, complexité

Dans le cas général, la réalisation pratique du générateur passe, comme dans l'exemple de la section précédente, par une étape préliminaire qui consiste à mettre la spécification

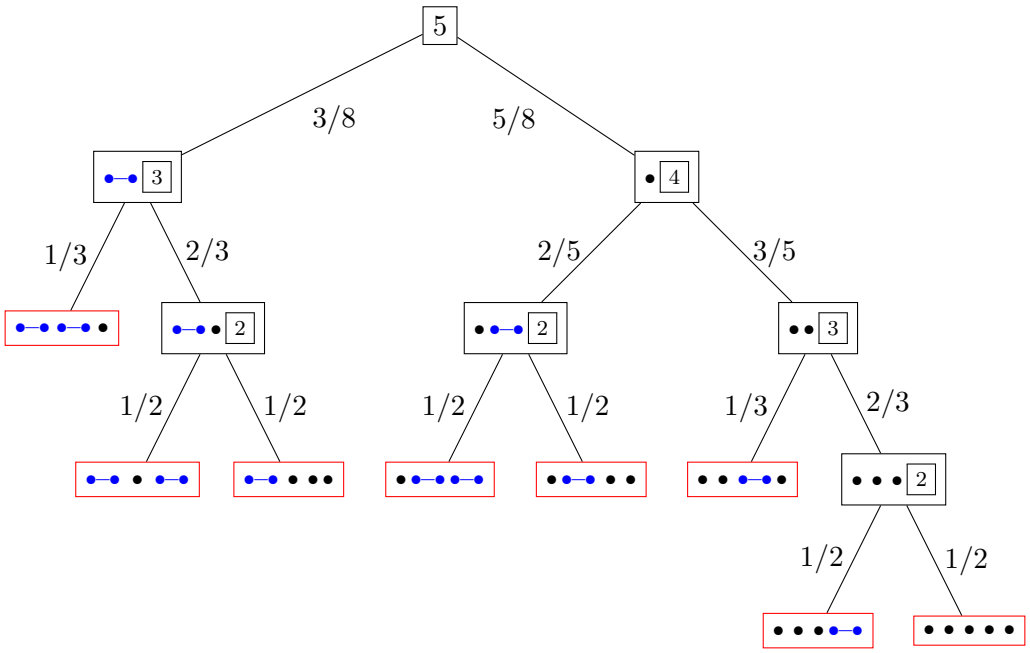


FIGURE 2.4 – Méthode récursive, arbre de dérivation de $\mathcal{C} = \mathcal{A} \times \mathcal{C} = \mathcal{Z} \times \mathcal{C} + \mathcal{Z}^2 \times \mathcal{C}$

de la classe combinatoire sous forme standard. Cette étape se résume essentiellement à exprimer les constructions admissibles à l'aide d'union et de produits binaires.

Dans le cas le plus général, les structures combinatoires de taille n sont engendrées avec une complexité en $\mathcal{O}(n^2)$. Le caractère quadratique est imposé par la règle produit $\mathcal{C} = \mathcal{A} \times \mathcal{B}$.

Il existe néanmoins des méthodes astucieuses, comme exposé dans [Phi94], pour parvenir à une complexité en $\mathcal{O}(n \log n)$ dans le pire des cas.

Le pré-traitement, destiné à calculer les coefficients de la série génératrice dont on a besoin, a lui aussi une complexité arithmétique quadratique [Fla90]. On trouve dans [Jor02] et [Piv08, chapitre 6] des moyens d'abaisser cette complexité arithmétique. En pratique les entiers mis en jeu sont cependant très grands et les résultats théoriques s'avèrent donc trop optimistes. Denise et Zimmermann montrent [DZ99] comment l'utilisation d'une arithmétique flottante permet de contourner cette difficulté.

2.3 Le principe des générateurs de Boltzmann

Les générateurs de Boltzmann naissent du passage de la notion de série formelle associée à la méthode précédente à l'interprétation analytique de ces objets. Si dans la méthode symbolique il s'agit de calculer les coefficients c_n dans :

$$C(z) = \sum_{n \in \mathbb{N}} c_n z^n.$$

La méthode de Boltzmann consiste à calculer $C(x)$ pour un nombre réel strictement positif x à l'intérieur du disque de convergence de la série. Évidemment, on ne peut plus alors "mettre la main" sur les structures de taille n pour n fixé. En revanche, on peut calculer l'espérance de produire des structures de tailles appartenant à un intervalle centré en n et dans certains cas, en utilisant un algorithme de rejet, il est même possible de produire des structures de taille n exactement.

Cette méthode permet de produire des structures récursives en grand nombre avec une complexité linéaire en la taille des données produites.

Il s'agit donc d'assouplir la contrainte sur la taille des objets engendrés. La taille des objets engendrés peut donc être considérée comme une variable aléatoire. Tous les objets de même taille doivent avoir la même probabilité de sortie. Tout l'art consiste à concentrer la taille des objets engendrés au plus proche de la taille cible afin qu'un éventuel algorithme de rejet soit efficace.

Dans la méthode de Boltzmann, les séries génératrices sont donc en particulier interprétées comme des fonctions analytiques.

Pour une classe combinatoire $\mathcal{C}$ dont la série génératrice est $C(z)$, on choisit donc un paramètre x (nombre réel strictement positif à l'intérieur du disque de convergence de la série). Le modèle (ordinaire) de Boltzmann de paramètre x assigne à tout objet γ de $\mathcal{C}$ la probabilité :

$$\boxed{\mathbb{P}(\gamma) = \frac{x^{|\gamma|}}{C(x)}.}$$

où $|\gamma|$ désigne la taille de γ.

Un générateur de Boltzmann $\Gamma C(x)$ pour $\mathcal{C}$ est un algorithme qui engendre des $\mathcal{C}$-structures suivant une distribution conforme à ce modèle. Le développement en série de $C(z)$ est inutile.

En particulier deux objets de même taille de $\mathcal{C}$ ont la même probabilité de tirage mais la taille des structures engendrées n'est pas (encore) contrôlée. On a :

$$\boxed{\mathbb{P}_x(N = n) = \sum_{|\gamma|=n} \frac{x^{|\gamma|}}{C(x)} = \frac{c_n x^n}{C(x)}.} \tag{2.1}$$

et :

$$\boxed{\mathbb{E}_x(N) = \frac{xC'(x)}{C(x)}, \quad \mathbb{E}_x(N^2) = \frac{x^2C''(x)}{C(x)} + \frac{xC'(x)}{C(x)}.} \tag{2.2}$$

Ce qui permet de calculer en particulier l'écart-type :

$$\boxed{\sigma(N) = \sqrt{\mathbb{E}_x(N^2) - \mathbb{E}_x(N)^2}.} \tag{2.3}$$

Classe combinatoire des mots binaires.

Pour la classe combinatoire des mots binaires, on a $W(z) = \frac{1}{1-2z}$ et pour tout $x \in]0, 1/2[$:

$$\boxed{\mathbb{P}_x(N = n) = (2x)^n(1 - 2x), \quad \mathbb{E}_x(N) = \frac{2x}{1 - 2x}.}$$

La figure fig 2.5 représente les courbes $n \mapsto \mathbb{P}_x(N = n)$ pour différentes valeurs de x.
La figure fig 2.6 représente la courbe $x \mapsto \mathbb{E}_x(N)$.

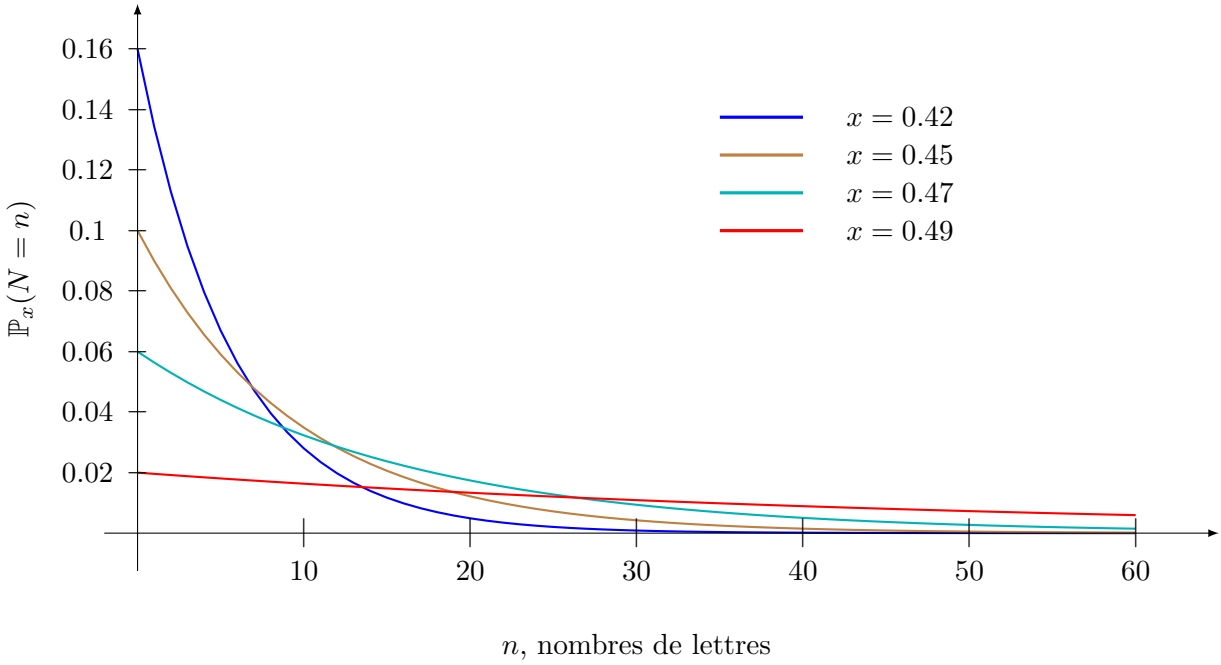


FIGURE 2.5 – $\mathbb{P}_x(N = n)$ en fonction de n pour différentes valeurs de x, mots binaires

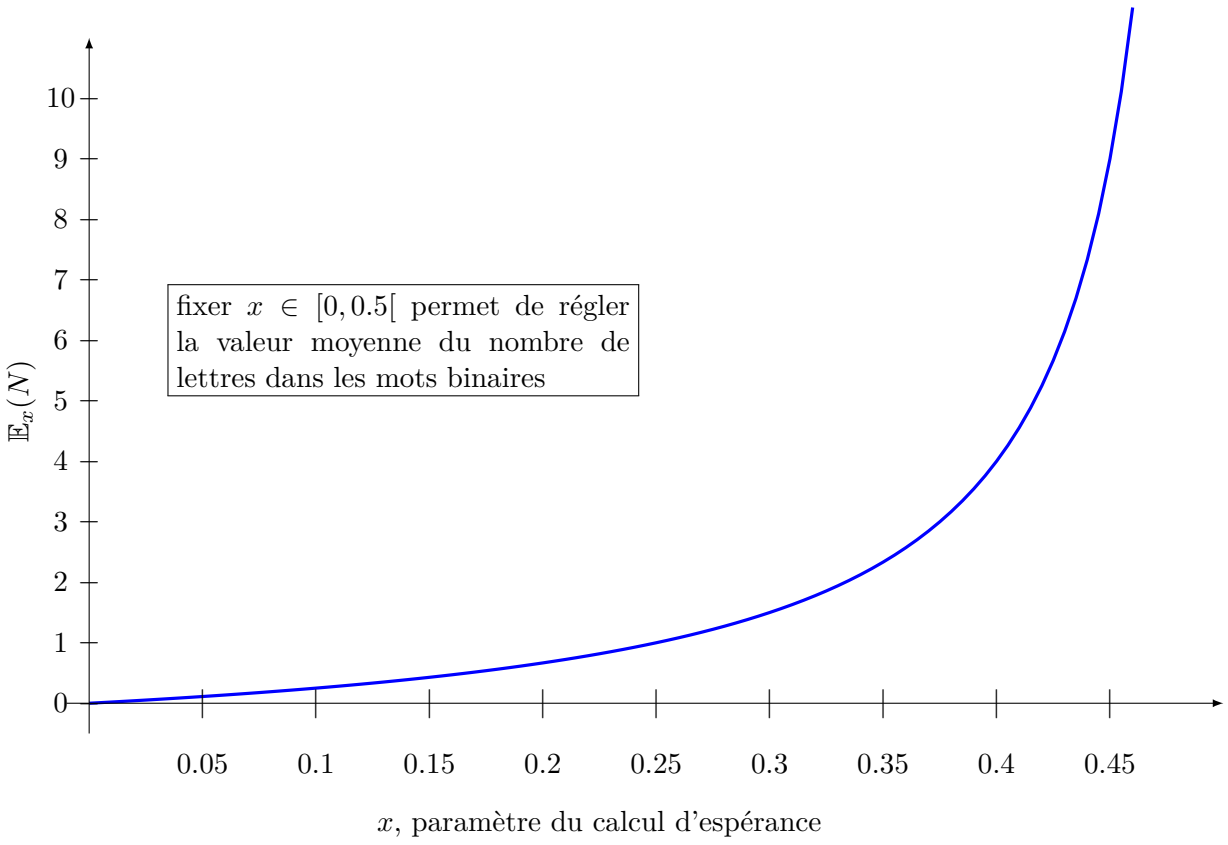


FIGURE 2.6 – $\mathbb{E}_x(N)$ en fonction de n de x, mots binaires

Classe combinatoire des partitions d'ensembles.

Pour la classe combinatoire des partitions d'ensembles, on a :

$$\mathcal{P}_e(z) = \exp(\exp(z) - 1)$$

et pour x réel positif fixé, on a :

$$\boxed{\mathbb{P}_x(N=n) = c_n x^n \exp\left(1-\exp(x)\right), \quad \mathbb{E}_x(N) = x\exp(x).}$$

La figure fig 2.7 représente les courbes $n \mapsto \mathbb{P}_x(N=n)$ pour différentes valeurs de n.

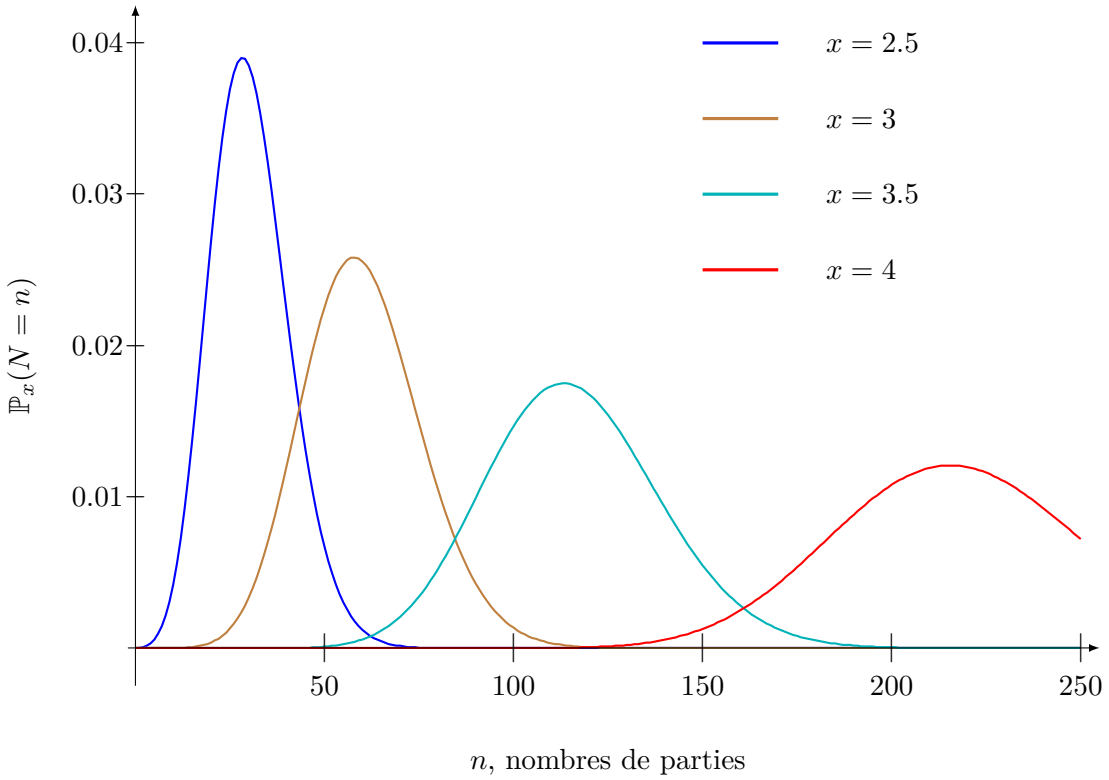


FIGURE 2.7 – $\mathbb{P}_x(N=n)$ en fonction de n pour différentes valeurs de x, partitions d'ensembles.

Comme dans le cas des mots binaires, la courbe représentant l'espérance en fonction du paramètre x donne des valeurs croissantes de 0 à $+\infty$. Le réglage du paramètre x permet de viser une taille pour les structures engendrées.

Construction d'un générateur simple dans le cas non-étiqueté.

Considérons une classe combinatoire $\mathcal{C}$ union disjointe des classes $\mathcal{A}$ et $\mathcal{B}$, on définit le générateur $\Gamma\mathcal{C}(x)$ par :

$$\boxed{\begin{array}{l}\Gamma\mathcal{C}(x) : \\ \qquad p_A \leftarrow \frac{A(x)}{C(x)} \\ \qquad \texttt{Si Bernoulli}(p_A)\ \texttt{Alors} \\ \qquad \texttt{renvoyer}\ \Gamma\mathcal{A}(x) \\ \qquad \texttt{sinon renvoyer}\ \Gamma\mathcal{B}(x)\end{array}} \quad , \tag{2.4}$$

où `Bernoulli(`p`)` désigne un tirage de Bernoulli de paramètre p.

Considérons une classe combinatoire $\mathcal{C}$ produit des classes $\mathcal{A}$ et $\mathcal{B}$, on définit le générateur $\Gamma\mathcal{C}(x)$ par :

$$\boxed{\begin{array}{l}\Gamma\mathcal{C}(x) : \\ \qquad \texttt{renvoyer}\ \ (\Gamma\mathcal{A}(x), \Gamma\mathcal{B}(x))\end{array}} \ . \tag{2.5}$$

Algorithm 2.4.1: Générateur à rejet associé à une classe $\mathcal{C}$

μC
Input: x, n, ε
Début
  repeat
    $\gamma := \Gamma C(x)$
  until $|\gamma| \in [\![n(1-\varepsilon),\ n(1+\varepsilon)]\!]$;
  return γ;

Appliquons ces deux idées très simples à la classe combinatoire $\mathcal{B}$ des arbres binaires. Cette classe est spécifiée par :

$$\mathcal{B} = \mathcal{Z} + \mathcal{B}^2.$$

On a donc immédiatement :

$$B(z) = z + B(z)^2.$$

On obtient :

$$B(x) = \frac{1 - \sqrt{1-4x}}{2}.$$

Le générateur de Boltzmann correspondant est donc défini par :

$\Gamma\mathcal{B}(x)$:
  $p \leftarrow \frac{x}{B(x)}$
  `Si Bernoulli(`p`) Alors`
  `renvoyer une feuille`
  `sinon renvoyer` $(\Gamma\mathcal{B}(x), \Gamma\mathcal{B}(x))$

2.4 Efficacité du rejet.

L'utilisation des générateurs de Boltzmann peut se faire en taille approximative ou en taille exacte. L'idée consiste, pour produire des objets de taille n ou des objets de taille t appartenant à l'intervalle $[\![n(1-\varepsilon), n(1+\varepsilon)]\!]$ (où ε est réel positif qui fixe l'intervalle de tolérance), à utiliser un algorithme de rejet qui ne conserve que les objets de tailles voulues.

Le principe du générateur à rejet $\mu C(x, n, \varepsilon)$, associé à un générateur de Boltzmann $\Gamma C(x)$, où :

- $x \in]0, \rho_C[$ (le demi-intervalle ouvert de convergence de $C(x)$),
- n est la taille ciblée par le générateur,
- $\varepsilon \geq 0$ est une tolérance relative,

est décrit par la figure 2.4.1.

Dans ce cadre précis, l'estimation de l'efficacité de l'algorithme est cruciale.

Un point précis sur le sujet est fait dans [DFLS04].

Dans la suite de cet exposé ne sont repris que des résultats liés à l'application pratique qui sera faite pour obtenir un automate à pile à partir de l'automate sous-jacent.

Pour un générateur de paramètre x associé à une fonction génératrice C, on note N la variable aléatoire liée à la taille des objets produits. En notant :

$$\mathbb{E}_x(N) = v_1(x), \quad \mathbb{E}_x\left(N^2\right) = v_2(x), \quad \mathbb{E}_x\left(N^2\right) - \mathbb{E}_x(N)^2 = \sigma^2(x).$$

on a :

$$v_1(x) = x\frac{C'(x)}{C(x)}, \quad v_2(x) = x^2\frac{C''(x)}{C(x)} + x\frac{C'(x)}{C(x)}, \quad \sigma(x) = \sqrt{v_2(x) - v_1^2(x)}.$$

$v_1(x)$ désigne donc l'espérance ou la moyenne de la variable aléatoire N, σ son écart-type.

Pour produire des objets de taille n quelconque efficacement il faut d'abord que tous les entiers n puissent constituer une valeur possible de l'espérance de la variable aléatoire. C'est le cas si :

$$\lim_{\substack{x\to\rho_C \\ x<\rho_C}} v_1(x) = +\infty. \tag{2.6}$$

Le critère 2.6 est appelé : **Condition de Valeur Moyenne**.

Un réglage naturel de $\mu C(x, n, \varepsilon)$ consiste évidemment à prendre pour valeur de x la solution x_n dans $]0, \rho_C[$ de l'équation :

$$n = x\frac{C'(x)}{C(x)}. \tag{2.7}$$

Le théorème 15 donne des conditions pour obtenir un algorithme de rejet efficace. Ce résultat est développé et commenté dans la section 6 (Exact-size and approximate-size sampling) de [DFLS04].

Théorème 15 (Conditions de la moyenne et de la variance)**.** *On considère $\mathcal{C}$ une classe combinatoire, ε une tolérance relative fixée.*

*Si la **Condition de la Valeur moyenne** 2.6 et la **Condition de Variance** 2.8 sont vérifiées :*

$$\lim_{\substack{x\to\rho_C \\ x<\rho_C}} \frac{\sigma(x)}{v_1(x)} = 0. \tag{2.8}$$

Alors, le générateur à rejet $\mu C(x_n, n, \varepsilon)$ (où x_n est solution de 2.7) réussit en un tirage avec une probabilité qui tend vers 1 lorsque n tend vers l'infini.

En particulier, si $\mathcal{C}$ est spécifiable, le coût du générateur en taille approximative est $\mathcal{O}(n)$.

2.5 Un résultat asymptotique utile

Nous présentons ici un résultat utile dans nos travaux. Il s'agit d'un des nombreux théorème d'analyse combinatoire permettant d'établir des résultats asymptotiques à partir de techniques de point col.
Ces techniques, comme le résultat, sont décrites dans [FS08].

Le résultat du théorème 16, [FS08, Theorem VIII.8] sera exploité dans la section 4.2.4 de ce travail.

Théorème 16. *On considère $C(z)$ une fonction génératrice ordinaire vérifiant :*

(1) *$C(z)$ est analytique en 0 et les coefficients de son développement sont tous strictement positifs,*
(2) *$C(0) \neq 0$,*
(3) *$C(z)$ est apériodique.*

Soit R le rayon de convergence de $C(z)$ et $T \stackrel{def}{=} \lim_{x \to R^-} x\frac{C'(x)}{C(x)}$.
Soient $\lambda \in]0, T[$ et ζ l'unique solution de l'équation d'inconnue x :

$$x\frac{C'(x)}{C(x)} = \lambda$$

Alors, pour $N \stackrel{def}{=} \lambda n$ un nombre entier, on a :

$$[z^N]C(z)^n = \frac{C(\zeta)^n}{\zeta^{N+1}\sqrt{2\pi n\xi}}(1 + o(1))$$

où $\xi = \frac{d^2}{dz^2}\left(\log C(z) - \lambda \log(z)\right)|_{z=\zeta}$.

Dans le théorème précédent T est appelé la *diffusion* de $C(z)$.

Chapitre 3

Chaînes de Markov, génération aléatoire, statistiques

Sommaire

3.1 Chaînes de Markov

Les chaînes de Markov constituent un domaine important et très vivace de la théorie des probabilités, elles proposent des outils pratiques statistiques qui s'appliquent dans des domaines variés [CG09, And99] [1]. Les méthodes de Monte-Carlo par chaînes de Markov sont exploitées dans l'approximation numérique, l'analyse numérique [Sai96]. On les retrouvent aussi communément en bioinformatique pour aborder par exemple des problèmes de phylogenèse en génétique [Alt04]. La linguistique s'intéresse à la parenté entre les langues et donc à une forme de phylogenèse qui donne lieu elle aussi à l'utilisation des chaînes de Markov [Nic08]. L'informatique propose également des champs variés d'application de cet outil. On retrouve ainsi les chaînes de Markov appliquées dans des problèmes de Machine Learning [And03]. Un domaine classique d'application des chaînes de Markov est celui des files d'attente [Bré13]. En informatique les questions de performance des serveurs sont directement liées à cette idée [Ram00].

Les chaînes de Markov jouent aussi un rôle important en informatique théorique à la fois pour mettre en œuvre des générateurs aléatoires ou pour proposer des algorithmes de comptage approximatifs. C'est cet aspect qui est utilisé dans ce travail.

Pour la génération aléatoire, le succès des chaînes de Markov s'explique par leur relative facilité d'adaptation aux différents problèmes. Les espaces d'états décrits par les chaînes

1. Les références sur les applications des chaînes de Markov sont destinées à pointer quelques exemples parmi d'autres fort nombreux et tout aussi pertinents. Ils n'ont aucun caractère exhaustif.

sont souvent très grands et la question critique classique consiste à se rapprocher, à une distance inférieure à un seuil requis, de la distribution stationnaire. Le nombre de pas pour atteindre ce seuil est appelé le *temps de mélange* de la chaîne.

La souplesse d'utilisation des chaînes de Markov a une contrepartie. Celle-ci réside souvent dans la grande difficulté à déterminer une estimation raisonnable du temps de mélange. Le chapitre 6 de ce travail, expose différentes méthodes d'appréciation de ce temps de mélange sur des exemples particuliers.

3.1.1 Introduction

Une chaîne de Markov finie décrit un processus qui gouverne aléatoirement le déplacement d'un élément d'un ensemble fini Ω à un autre élément de Ω. Une vision grossière de ce processus consiste à dire que si x est un élément de l'ensemble Ω alors, en partant de x, la position suivante sera déterminée par une probabilité $P(x, \cdot)$ définie sur l'ensemble Ω. La définition 17 précise formellement les choses.

Définition 17 (chaîne de Markov)**.** *On considère Ω un ensemble fini non vide, une chaîne de Markov (finie) d'espace d'états Ω et de matrice de transition M, est une suite de variables aléatoires $(X_t)_{t\in\mathbb{N}}$ à valeurs dans Ω telle que pour tous $x, y \in \Omega$, pour tout $t \in \mathbb{N}\backslash\{0\}$ et tout événement de la forme :*

$$\bigcap_{s=0}^{t-1} (X_s = x_s),$$

tel que

$$\mathbb{P}\left((X_t = x) \cap \bigcap_{s=0}^{t-1} (X_s = x_s)\right) > 0,$$

alors :

$$\mathbb{P}\left((X_{t+1} = y) \,|\, (X_t = x) \cap \bigcap_{s=0}^{t-1} (X_s = x_s)\right) = \mathbb{P}\left((X_{t+1} = y) \,|\, (X_t = x)\right) = M(x, y). \quad (3.1)$$

L'équation 3.1 est appelée *propriété de Markov*. On traduit souvent cette propriété en disant qu'une chaîne de Markov est *sans mémoire* pour exprimer le fait que la probabilité conditionnelle de transiter de l'état x à l'état y entre les dates t et $t+1$ ne dépend pas des états x_0, $x-1$, …, x_{t-1} visités aux dates antérieures.

Dans toute la suite, les espaces d'états considérés sont finis, on écrira donc simplement *chaîne de Markov* pour *chaîne de Markov finie*. De même, quoique la terminologie soit impropre, si M est la matrice de transition associée à une chaîne de Markov, on s'autorisera à écrire *la chaîne de Markov M* plutôt que *la chaîne de Markov de matrice de transition M*.

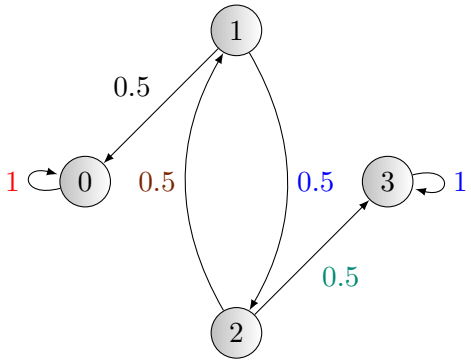
La matrice M décrit donc entièrement le processus (modulo le point de départ). C'est une matrice carrée de taille $|\Omega|$ dont tous les coefficients sont positifs. Il est naturel d'indicer les coefficients de la matrice M par les éléments de Ω et dans la suite on notera donc $M(x, y)$ le coefficient de la matrice M qui caractérise la probabilité de transition de l'état x à l'état y, en parlant du coefficient de M de la x-ième ligne et y-ième colonne.
La matrice M vérifie par ailleurs :

$$\forall x \in \Omega, \quad \sum_{y\in\Omega} M(x, y) = 1,$$

elle est donc stochastique.

On peut associer un graphe orienté G à une chaîne de Markov sur un espace d'états Ω. L'ensemble des sommets de G est Ω, et pour tout $(x, y) \in \Omega^2$, (x, y) est un arc de G si et seulement si $M(x, y) > 0$.

La figure 3.1 illustre ces idées par l'exemple classique de la *ruine du joueur*. Dans l'exemple choisi, un joueur joue à pile ou face avec 1 euro en poche jusqu'à avoir perdu ou avoir 3 euros en poche. À chaque étape, s'il gagne il remporte un euro et s'il perd il en perd un.



$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

FIGURE 3.1 – Graphe et matrice associés à la ruine du joueur

On peut noter aussi que pour tout nombre entier naturel t, $M^t(x, y)$, le coefficient de la x-ième ligne et y-ième colonne de M^t est la probabilité (sachant que l'état courant est x) de passer de l'état x à l'état y en t pas. Dire qu'il existe un chemin de longueur t de x à y dans le graphe sous-jacent associé à la chaîne de Markov considérée est donc équivalent à $M^t(x, y) > 0$. En particulier, il existe dans G un cycle de longueur t passant par x si et seulement si $M^t(x, x) > 0$. Les définitions 18, 19 et 22 proposent donc deux façons équivalentes de caractériser les mêmes notions.

Définition 18 (période d'un élément de Ω)**.** *On considère x un élément de l'espace d'états Ω d'une chaîne de Markov M.*

On appelle période de x, et on note $\tau(x)$, le plus grand commun diviseur de l'ensemble

$$\Big\{t \in \mathbb{N}\backslash\{0\},\ M^t(x, x) > 0\Big\}.$$

La période de x est donc aussi le plus grand commun diviseur de l'ensemble des longueurs des cycles de G passant par x.

Remarque.
Dans l'exemple de la figure 3.1, les sommets 0 et 3 sont de période 1 alors que les sommets 1 et 2 sont de période 2.

Définition 19 (irréductibilité)**.** *On considère M la matrice de transition et G le graphe sous-jacent à une chaîne de Markov sur un espace d'états Ω.*

On dit que la chaîne M est irréductible si pour tous les états x et y de Ω il existe un nombre entier naturel t tel que $M^t(x, y) > 0$.

La chaîne M est donc irréductible si le graphe G est fortement connexe.

Remarque.
La chaîne de la figure 3.1 n'est pas irréductible puisque les sommets 0 et 3 sont des états "puits". Il est impossible de rejoindre un autre sommet en partant d'un de ces deux états.

Proposition 20 (périodicité)**.** *On considère une chaîne de Markov M irréductible. Alors tous les états de l'espace d'états Ω associé, ont la même période.*

Définition 21 (période d'une chaîne irréductible)**.** *On considère M une chaîne de Markov irréductible. On appelle période de M et on note $\tau(M)$ la période commune de tous les éléments de l'espace des états associé à M.*

Définition 22 (apériodicité)**.** *On considère M une chaîne de Markov irréductible. On dit que M est apériodique si $\tau(M) = 1$. On dit que la chaîne est périodique dans le cas contraire.*
Une chaîne est donc apériodique si le plus grand commun diviseur de l'ensemble des longueurs cycles passant par un élément quelconque de la chaîne est égale à 1.

La proposition 23 exprime l'idée que si une chaîne de Markov est irréductible et apériodique, il est alors possible de rejoindre un état à un autre par un chemin de longueur r. La longueur de ce chemin ne dépend pas des deux états à relier.

Proposition 23 (irréductibilité et apériodicité)**.** *Si M est une chaîne de Markov irréductible et apériodique, il existe un nombre entier strictement positif r tel que pour tout couple (x, y) de l'espace d'états associés $M^r(x, y) > 0$.*

Exemple du dé à 6 faces Dans cette introduction on se propose d'illustrer les notions introduites par un modèle très simple de chaîne de Markov. Le but consiste dans la suite à utiliser cet outil pour construire un générateur aléatoire. On propose donc ici un exemple jouet, qui nous suivra dans la suite, pour illustrer la façon dont on pourrait simuler un lancement de dé à 6 faces avec une chaîne de Markov.

L'espace des états est donc naturellement défini par $\Omega \overset{\text{def}}{=} \{1, 2, 3, 4, 5, 6\}$ et la matrice de transition M par :

$$\forall (i, j) \in \{1, 2, 3, 4, 5, 6\},\ M(i, j) = \begin{cases} \frac{1}{2} \text{ si } j = i + 1[6] \\ \frac{1}{2} \text{ si } j = i - 1[6] \\ 0 \text{ sinon} \end{cases} .$$

La figure 3.2 montre le graphe associé à cette chaîne de Markov que l'on appelle marche aléatoire sur le 6-cycle (bien souvent on remplace l'ensemble $\{1, 2, 3, 4, 5, 6\}$ par $\{0, 1, 2, 3, 4, 5\}$ que l'on peut alors identifier au groupe $\mathbb{Z}/6\mathbb{Z}$.

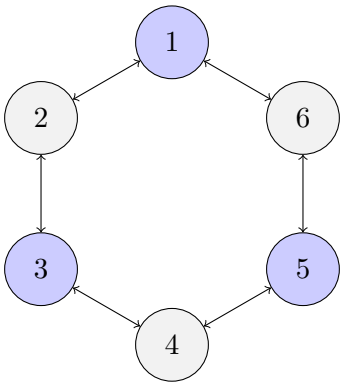


FIGURE 3.2 – Marche aléatoire sur le 6-cycle

Après chaque mouvement dans la marche aléatoire sur le 6-cycle, on change la parité de l'état. Il est par ailleurs clair que cette chaîne est irréductible et périodique de période 2.

L'idée générale des méthodes de Monte-Carlo par chaîne de Markov consiste à partir d'un sommet initial du graphe associé à la chaîne, à réaliser n mouvements et à retourner l'état obtenu après ces n mouvements dans la chaîne.
Pour la marche aléatoire sur le 6-cycle que nous venons de décrire cette idée ne fonctionne pas pour obtenir un générateur uniforme, dans la mesure où en partant par exemple d'un sommet pair le sommet obtenu au bout de n pas aura la parité de n. Dit autrement, la suite de matrices $(M^t)_{t\in\mathbb{N}}$ ne peut pas converger.

La figure 3.3 donne quelques valeurs des puissances de la matrice de transition associée à la marche aléatoire sur le 6-cycle.

$$M = \begin{pmatrix} 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 \end{pmatrix}, \quad M^2 = \begin{pmatrix} 1/2 & 0 & 1/4 & 0 & 1/4 & 0 \\ 0 & 1/2 & 0 & 1/4 & 0 & 1/4 \\ 1/4 & 0 & 1/2 & 0 & 1/4 & 0 \\ 0 & 1/4 & 0 & 1/2 & 0 & 1/4 \\ 1/4 & 0 & 1/4 & 0 & 1/2 & 0 \\ 0 & 1/4 & 0 & 1/4 & 0 & 1/2 \end{pmatrix},$$

$$M^3 = \begin{pmatrix} 0 & 3/8 & 0 & 1/4 & 0 & 3/8 \\ 3/8 & 0 & 3/8 & 0 & 1/4 & 0 \\ 0 & 3/8 & 0 & 3/8 & 0 & 1/4 \\ 1/4 & 0 & 3/8 & 0 & 3/8 & 0 \\ 0 & 1/4 & 0 & 3/8 & 0 & 3/8 \\ 3/8 & 0 & 1/4 & 0 & 3/8 & 0 \end{pmatrix}, \quad M^4 = \begin{pmatrix} 3/8 & 0 & 5/16 & 0 & 5/16 & 0 \\ 0 & 3/8 & 0 & 5/16 & 0 & 5/16 \\ 5/16 & 0 & 3/8 & 0 & 5/16 & 0 \\ 0 & 5/16 & 0 & 3/8 & 0 & 5/16 \\ 5/16 & 0 & 5/16 & 0 & 3/8 & 0 \\ 0 & 5/16 & 0 & 5/16 & 0 & 3/8 \end{pmatrix},$$

$$M^5 = \begin{pmatrix} 0 & 11/32 & 0 & 5/16 & 0 & 11/32 \\ 11/32 & 0 & 11/32 & 0 & 5/16 & 0 \\ 0 & 11/32 & 0 & 11/32 & 0 & 5/16 \\ 5/16 & 0 & 11/32 & 0 & 11/32 & 0 \\ 0 & 5/16 & 0 & 11/32 & 0 & 11/32 \\ 11/32 & 0 & 5/16 & 0 & 11/32 & 0 \end{pmatrix}, \quad M^6 = \begin{pmatrix} 11/32 & 0 & 21/64 & 0 & 21/64 & 0 \\ 0 & 11/32 & 0 & 21/64 & 0 & 21/64 \\ 21/64 & 0 & 11/32 & 0 & 21/64 & 0 \\ 0 & 21/64 & 0 & 11/32 & 0 & 21/64 \\ 21/64 & 0 & 21/64 & 0 & 11/32 & 0 \\ 0 & 21/64 & 0 & 21/64 & 0 & 11/32 \end{pmatrix}.$$

FIGURE 3.3 – Différentes puissances de la matrice de transition associée au 6-cycle

On démontre facilement que :

$$\lim_{t\to\infty} M^{2t} = \begin{pmatrix} 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \end{pmatrix}, \quad \lim_{t\to\infty} M^{2t+1} = \begin{pmatrix} 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \end{pmatrix}.$$

En appelant M_0 la première limite et M_1 la seconde, on a :

- Pour tout $v_1 \in \{(1,0,0,0,0,0), (0,0,1,0,0,0), (0,0,0,0,1,0)\}$:

$$v_1M_0 = (1/3, 0, 1/3, 0, 1/3, 0),\ v_1M_1 = (0, 1/3, 0, 1/3, 0, 1/3)$$

- Pour tout $v_0 \in \{(0,1,0,0,0,0), (0,0,0,1,0,0), (0,0,0,0,0,1)\}$:

$$v_0M_0 = (0, 1/3, 0, 1/3, 0, 1/3),\ v_0M_1 = (1/3, 0, 1/3, 0, 1/3, 0)$$

En clair, en partant d'un état impair (resp. pair) lorsque le nombre de mouvements dans la chaîne est pair et tend vers l'infini, on obtient en fin de parcours un état impair (resp. pair) de façon équiprobable sur les 3 états impairs (resp. pairs).

On obtient le même type de résultats lorsque le nombre de mouvements dans la chaîne est impair et tend vers l'infini.

Un moyen simple de palier les soucis liés à cette situation consiste à proposer la version dite *paresseuse* de la même chaîne. L'idée consiste à rester sur le même sommet avec la probabilité $\frac{1}{2}$ et de décider des autres mouvements possibles dans la chaîne de départ avec les mêmes probabilités multipliées par le même facteur $\frac{1}{2}$. En clair on tire au sort de façon équitable si au pas suivant on reste sur le même état ou non. Cela revient juste à remplacer la matrice de transition M par la matrice $M' \stackrel{\text{def}}{=} \frac{1}{2}(M + I)$, où I est la matrice carrée identité de taille $|\Omega|$.

La version paresseuse de la marche aléatoire sur un 6-cycle est illustrée par la figure 3.4.

Le point essentiel consiste à remarquer que la version *paresseuse* d'une chaîne de Markov irréductible est par construction apériodique.

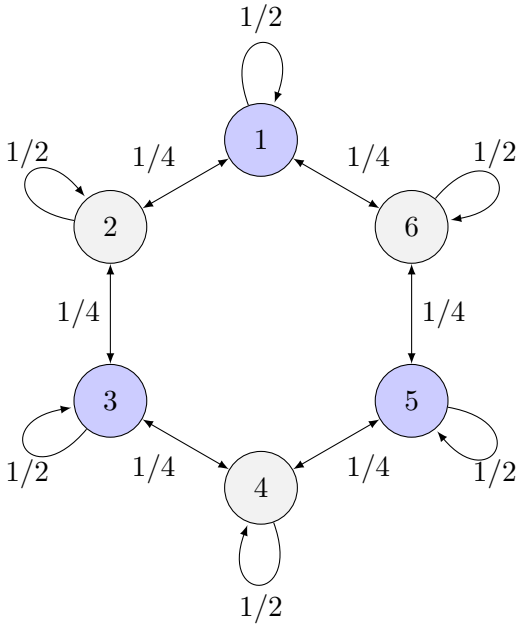


FIGURE 3.4 – Version paresseuse de la marche aléatoire sur un 6-cycle

Dans le cas d'une marche aléatoire sur un 6-cycle, la figure 3.5 donne quelques valeurs de M'^n. On observe que les puissances successives de la matrice correspondant à la version *paresseuse* de la chaîne semblent converger vers la matrice :

$$\frac{1}{6}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} = \frac{1}{6} J_6.$$

Multiplication à droite ou à gauche par un vecteur unidimensionnel. Il faut noter par ailleurs que les matrices M ou M' conduisent à deux interprétations suivant que l'on considère leur multiplication à gauche par une matrice ligne ou à droite par une matrice colonne.

On peut considérer une matrice ligne μ comme associée à une loi de probabilité sur les 6 états possibles de l'espace des états. Ainsi $\mu = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ correspond à une

$$M' = \begin{pmatrix} 1/2 & 1/4 & 0 & 0 & 0 & 1/4 \\ 1/4 & 1/2 & 1/4 & 0 & 0 & 0 \\ 0 & 1/4 & 1/2 & 1/4 & 0 & 0 \\ 0 & 0 & 1/4 & 1/2 & 1/4 & 0 \\ 0 & 0 & 0 & 1/4 & 1/2 & 1/4 \\ 1/4 & 0 & 0 & 0 & 1/4 & 1/2 \end{pmatrix} = \begin{pmatrix} 0.5 & 0.25 & 0 & 0 & 0 & 0.25 \\ 0.25 & 0.5 & 0.25 & 0 & 0 & 0 \\ 0 & 0.25 & 0.5 & 0.25 & 0 & 0 \\ 0 & 0 & 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0 & 0 & 0.25 & 0.5 & 0.25 \\ 0.25 & 0 & 0 & 0 & 0.25 & 0.5 \end{pmatrix},$$

$$M'^2 = \begin{pmatrix} 0.375 & 0.25 & 0.0625 & 0 & 0.0625 & 0.25 \\ 0.25 & 0.375 & 0.25 & 0.0625 & 0 & 0.0625 \\ 0.0625 & 0.25 & 0.375 & 0.25 & 0.0625 & 0 \\ 0 & 0.0625 & 0.25 & 0.375 & 0.25 & 0.0625 \\ 0.0625 & 0 & 0.0625 & 0.25 & 0.375 & 0.25 \\ 0.25 & 0.0625 & 0 & 0.0625 & 0.25 & 0.375 \end{pmatrix},$$

$$M'^3 = \begin{pmatrix} 0.3125 & 0.234375 & 0.09375 & 0.03125 & 0.09375 & 0.234375 \\ 0.234375 & 0.3125 & 0.234375 & 0.09375 & 0.03125 & 0.09375 \\ 0.09375 & 0.234375 & 0.3125 & 0.234375 & 0.09375 & 0.03125 \\ 0.03125 & 0.09375 & 0.234375 & 0.3125 & 0.234375 & 0.09375 \\ 0.09375 & 0.03125 & 0.09375 & 0.234375 & 0.3125 & 0.234375 \\ 0.234375 & 0.09375 & 0.03125 & 0.09375 & 0.234375 & 0.3125 \end{pmatrix},$$

$$M'^4 = \begin{pmatrix} 0.2734375 & 0.21875 & 0.1132812 & 0.0625 & 0.1132812 & 0.21875 \\ 0.21875 & 0.2734375 & 0.21875 & 0.1132812 & 0.0625 & 0.1132812 \\ 0.1132812 & 0.21875 & 0.2734375 & 0.21875 & 0.1132812 & 0.0625 \\ 0.0625 & 0.1132812 & 0.21875 & 0.2734375 & 0.21875 & 0.1132812 \\ 0.1132812 & 0.0625 & 0.1132812 & 0.21875 & 0.2734375 & 0.21875 \\ 0.21875 & 0.1132812 & 0.0625 & 0.1132812 & 0.21875 & 0.2734375 \end{pmatrix},$$

$$M'^9 = \begin{pmatrix} 0.1916962 & 0.1791801 & 0.1541519 & 0.1416397 & 0.1541519 & 0.1791801 \\ 0.1791801 & 0.1916962 & 0.1791801 & 0.1541519 & 0.1416397 & 0.1541519 \\ 0.1541519 & 0.1791801 & 0.1916962 & 0.1791801 & 0.1541519 & 0.1416397 \\ 0.1416397 & 0.1541519 & 0.1791801 & 0.1916962 & 0.1791801 & 0.1541519 \\ 0.1541519 & 0.1416397 & 0.1541519 & 0.1791801 & 0.1916962 & 0.1791801 \\ 0.1791801 & 0.1541519 & 0.1416397 & 0.1541519 & 0.1791801 & 0.1916962 \end{pmatrix},$$

$$M'^{20} = \begin{pmatrix} 0.1677237 & 0.1671952 & 0.1661381 & 0.1656096 & 0.1661381 & 0.1671952 \\ 0.1671952 & 0.1677237 & 0.1671952 & 0.1661381 & 0.1656096 & 0.1661381 \\ 0.1661381 & 0.1671952 & 0.1677237 & 0.1671952 & 0.1661381 & 0.1656096 \\ 0.1656096 & 0.1661381 & 0.1671952 & 0.1677237 & 0.1671952 & 0.1661381 \\ 0.1661381 & 0.1656096 & 0.1661381 & 0.1671952 & 0.1677237 & 0.1671952 \\ 0.1671952 & 0.1661381 & 0.1656096 & 0.1661381 & 0.1671952 & 0.1677237 \end{pmatrix}.$$

FIGURE 3.5 – Différentes puissances de la matrice de transition associée au 6-cycle, version *paresseuse*

situation dans laquelle on se trouve à l'étape t à l'état 1 avec la probabilité 1. On a alors :

$$\mu M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1/2 & 0 & 0 & 0 & 1/2 \end{pmatrix}.$$

Le résultat du calcul donne la répartition au temps $t+1$, où l'on se trouve avec la probabilité $\frac{1}{2}$ en 2 ou 6.

Le fait que M' converge vers $\frac{1}{6}J_6$ s'interprète en disant que quelle que soit la distribution de départ, on tend vers la distribution $\frac{1}{6}\begin{pmatrix}1 & 1 & 1 & 1 & 1 & 1\end{pmatrix}$ qui est la distribution uniforme.

La multiplication de M ou M' par un vecteur colonne peut s'interpréter comme une espérance. En effet si le i-ème coefficient du vecteur colonne représente les valeurs d'une fonction f définie sur l'ensemble des états en i, on a :

$$M \cdot f(i) = \sum_{j=1}^{6} M_{i,j} f(j) = \sum_{j=1}^{6} f(j) \mathbb{P}\left(X_{t+1} = j | X_t = i\right).$$

Il s'agit donc de l'espérance de la valeur de la fonction f à l'étape $t+1$ sachant que l'on se trouve à l'état i à l'étape t.

Il faut noter que si la matrice M est symétrique la multiplication à gauche par un vecteur ligne u donne le même résultat que ${}^t(M({}^t u))$.

La section suivante expose quelques points indispensables de théorie.

3.1.2 Un peu de théorie

Distributions stationnaires

On considère π une loi de probabilité sur Ω, M la matrice de transition associée à une chaîne de Markov. On dit que π est une *distribution stationnaire* ou une *probabilité invariante* lorsque :

$$\pi = \pi M. \tag{3.2}$$

Il faut voir dans cette égalité la distribution π comme un vecteur de probabilité. L'égalité 3.2 traduit le fait que le vecteur de probabilité π est un vecteur propre gauche de la matrice M associé à la valeur propre 1.

La terminologie adoptée est claire. En effet, si la chaîne démarre à la date $t = 0$ dans une distribution qui correspond à une distribution stationnaire, c'est-à-dire que $\mu_0 = \pi$ alors pour toute date t, on a $\mu_t = \pi$.

Pour illustrer sur un exemple simple cette notion de distribution stationnaire ou probabilité invariante on considère un espace Ω à trois états $\Omega \stackrel{\text{def}}{=} \{T, R, D\}$ qui illustre les 3 états dans lesquels peut se trouver un thésard (Travailler, se Restaurer, Dormir). La matrice de transition associée à la chaîne de Markov qui modélise les passages d'un état à un autre est la suivante :

$$M = \begin{pmatrix} 0.6 & 0.3 & 0.1 \\ 0.5 & 0.3 & 0.2 \\ 0.1 & 0.7 & 0.2 \end{pmatrix}.$$

Pour déterminer une distribution stationnaire on normalise les vecteurs du noyau de ${}^t M - I$, où I est la matrice identité 3 lignes, 3 colonnes. En arrondissant, on trouve une seule distribution stationnaire :

$$\pi = (0.49, 0.36, 0.15).$$

On a par ailleurs :

$$M^3 = \begin{pmatrix} 0.496 & 0.356 & 0.148 \\ 0.487 & 0.36 & 0.153 \\ 0.467 & 0.376 & 0.157 \end{pmatrix}.$$

Au cours du temps, quelle que soit la configuration initiale μ_0, la probabilité de trouver l'étudiant dans l'un des 3 états T, R et D tend vers la seule distribution stationnaire possible.
La suite formalise ces observations.

Distribution stationnaire, conditions d'unicité, convergence

Les résultats théoriques qui permettent d'établir l'existence et l'unicité d'une probabilité invariante s'appuient sur le théorème de Perron-Frobenius. Nous commençons par rappeler quelques définitions simples :

- Si A est une matrice carrée réelle, on appelle *spectre de* A et on note $\sigma(A)$ l'ensemble des valeurs propres de A.
  On appelle *rayon spectral de* A et on note $r(A)$ le nombre réel positif défini par :

$$r(A) \stackrel{\text{def}}{=} \max_{\lambda \in \sigma(A)} |\lambda|.$$

  On dit d'une valeur propre λ de A qu'elle est *simple* si le sous-espace propre associé est de dimension 1.
  On dit d'une valeur propre λ de A qu'elle est dominante si :

$$\forall \alpha \in \sigma(A) \backslash \{\lambda\}, \quad |\lambda| > |\alpha|.$$

- On dit d'une matrice A qu'elle est positive (resp. strictement positive) si tous ses coefficients sont positifs (resp. strictement positifs). Si c'est le cas, on note $A \geqslant 0$ (resp. $A > 0$). On définit de la même façon les notions de vecteur positif ou strictement positif.
- On dit d'une matrice A carrée, réelle **positive** qu'elle est *primitive* s'il existe un nombre entier naturel k tel que $A^k > 0$.
  On dit que A est *irréductible* si pour tout couple d'indices (i, j) de ligne et de colonne de A, il existe un nombre entier naturel (qui dépend donc a priori du couple (i, j)) tel que le coefficient de la i-ième ligne et j-ième colonne de A^k soit strictement positif.

Il est clair qu'une matrice primitive est également irréductible, la réciproque est fausse en général. En effet, si l'on considère la matrice A définie par :

$$A \stackrel{\text{def}}{=} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

On a :

$$\forall k \in \mathbb{N}, \quad A^{2k} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad A^{2k+1} = A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

La matrice A est donc irréductible elle n'est en revanche pas primitive. En interprétant A comme une matrice stochastique, on perçoit clairement que ce résultat est lié au fait qu'aucune transition d'un état vers lui même n'est possible. On verra que pour la version paresseuse de A les deux propriétés sont vérifiées.

On peut aussi noter que A admet deux valeurs propres simples 1 et -1 et que A n'admet donc pas de valeur propre dominante.

On démontre qu'une matrice A est positive si et seulement si pour tout vecteur x positif, le vecteur Ax est positif. De même A est strictement positive si et seulement si pour tout vecteur x strictement positif, le vecteur Ax est un vecteur strictement positif.

Théorème 24 (Perron-Frobenius)**.** *On considère une matrice carrée* ***positive*** $A = (a_{ij}) \in \mathcal{M}_n(\mathbb{R})$.

- *(**Perron**) : Si A est primitive son rayon spectral $r(A)$ est strictement positif et est une valeur propre **dominante** simple de A. Le sous-espace propre associé est dirigé par un vecteur de probabilité strictement positif.*
- *(**Frobenius**) : Si A est irréductible son rayon spectral $r(A)$ est strictement positif et est une valeur propre simple de A. Le sous-espace propre associé est dirigé par un vecteur de probabilité strictement positif.*

Si M est une matrice associée à une chaîne de Markov, M est stochastique et par là même positive. Chercher une probabilité invariante pour M consiste à chercher un vecteur propre de ${}^tM$.
On note aussi que la définition du caractère irréductible d'une matrice positive coïncide alors avec celle de chaîne de Markov irréductible.

Puisque la matrice M est stochastique, le vecteur $(1, \dots 1)$ est un vecteur propre de M. Le fait que les matrices M et ${}^tM$ aient même spectre garantit que 1 est également valeur propre de ${}^tM$.

On démontre par ailleurs facilement que le rayon spectral de M (ou de ${}^tM$) est égal à 1.

Si la matrice M est irréductible, le théorème de Perron-Frobenius assure donc qu'il existe une unique probabilité invariante π et que cette distribution stationnaire est à coefficients strictement positifs.

Un autre résultat essentiel porte sur la convergence de la suite de vecteurs $(\mu_0 M^k)_{k\in\mathbb{N}}$ pour une distribution initiale μ_0. Dans le théorème qui suit $\|x\|_1$ désigne la norme 1 du vecteur x, c'est-à-dire que :

$$\|x\|_1 = \sum_{k=1}^{n} |x_i|.$$

où $(x_1, \dots, x_n)$ est le n-uplet des coordonnées de x.

Théorème 25 (convergence)**.** *Si M est une matrice stochastique **primitive** admettant la distribution stationnaire π, pour toute distribution initiale μ_0 on a alors :*

$$\lim_{k\to\infty} \left\|\mu_0 M^k - \pi\right\|_1 = 0.$$

Plus précisément, la convergence est exponentielle puisque :

$$\left\|\mu_0 M^k - \pi\right\|_1 \underset{k\to\infty}{=} O\left(\rho^k\right),$$

où

$$\rho \overset{def}{=} \max\left\{|\alpha|,\, \alpha \in \sigma(M)\backslash\{1\}\right\}.$$

Les résultats éclairent donc les résultats observés sur la matrice stochastique

$$M = \begin{pmatrix} 0.6 & 0.3 & 0.1 \\ 0.5 & 0.3 & 0.2 \\ 0.1 & 0.7 & 0.2 \end{pmatrix}.$$

La vitesse de convergence est reliée de manière clef à la deuxième valeur propre dominante. Dans l'exemple jouet on a (en donnant des valeurs approchées des valeurs propres différentes de 1) :

$$\sigma(M) = \{1, 0.256, -0.156\},$$

ce qui explique la convergence très rapide M^k.

Dans le cas de la marche aléatoire paresseuse sur un 6-cycle on a :

$$\sigma = \left\{1, \frac{3}{4}, \frac{1}{4}, 0\right\}.$$

Les sous-espaces propres associés à $\frac{3}{4}$ et $\frac{1}{4}$ sont de dimension 2.

Deux points importants restent à souligner ici :

- Si M est une matrice stochastique **symétrique** alors M et ${}^tM$ ont non seulement le même spectre mais aussi les mêmes sous-espaces propres. Ainsi, puisque le vecteur $(1, \ldots, 1)$ dirige le sous-espace propre associé à la valeur propre 1, la distribution stationnaire est la distribution **uniforme**.
- Si M est irréductible et apériodique le résultat de la proposition 23 assure que M est primitive. Les hypothèses du théorème 25 sont alors vérifiées. On remplace d'ailleurs souvent l'hypothèse qui précise que M est primitive par celle qui précise que M est apériodique et on dit que M est *ergodique* lorsqu'elle est à la fois irréductible et apériodique. On peut noter que c'est l'hypothèse d'apériodicité qui permet de définir $\rho < 1$ et que cette hypothèse est vérifiée pour les versions paresseuses des chaînes de Markov.

3.1.3 Distance en variation totale, temps de mélange, couplage

Le théorème 25 de la section précédente donne un résultat de convergence des chaînes ergodiques pour une métrique particulière et en s'appuyant sur le spectre de la matrice de transition M. Cette approche est évidemment difficile à mettre en œuvre pour des espaces d'états Ω de grande taille pour lesquels le calcul du spectre de M, ou de simples majorations des valeurs propres différentes de 1 s'avèrent une entreprise très ardue. Une idée consiste donc à envisager différentes métriques pour étudier les questions de convergence, [Gib02] propose une étude sur ce sujet. Une des métriques les plus commodes, parce qu'elles offre des outils intéressants, est la *distance en variation totale*. Nous présentons brièvement cette métrique.

Distance en variation totale

La distance en variation totale est une distance sur l'ensemble des lois de probabilité sur un même ensemble d'états fini Ω. Si μ et ν sont deux lois de probabilité sur Ω la distance en variation totale entre μ et ν est le maximum de la distance entre les probabilités du même événement A. Plus formellement, en notant $\|\mu - \nu\|_{TV}$ la distance en variation totale entre μ et ν, on a :

$$\|\mu - \nu\|_{TV} \stackrel{\text{def}}{=} \max_{A \subset \Omega} |\mu(A) - \nu(A)|.$$

On montre que cette définition conduit à des re-formulations équivalentes :

$$\begin{aligned} \|\mu - \nu\|_{TV} &= \frac{1}{2} \sum_{x \in \Omega} |\mu(x) - \nu(x)| \\ \|\mu - \nu\|_{TV} &= \sum_{\substack{x \in \Omega \\ \mu(x) \geqslant \nu(x)}} (\mu(x) - \nu(x)). \end{aligned}$$

Une autre façon encore de définir la notion de distance en variation totale consiste à considérer, pour deux lois de probabilité μ et ν sur l'espace Ω, tous les couples de variables aléatoires (X, Y) tels que les lois marginales soient égales à μ et ν. On dit qu'un tel couple de variables aléatoires est un *couplage* des lois μ et ν. La distance en variation totale entre μ et ν est la borne inférieure de la mesure de l'événement $(X \neq Y)$:

$$\|\mu - \nu\|_{TV} = \inf \{\mathbb{P}(X \neq Y), \quad (X, Y) \text{ couplage de } \mu \text{ et } \nu\}. \tag{3.3}$$

On peut en fait démontrer que la borne inférieure de l'équation 3.3 est atteinte et constitue donc un minimum. Une couplage pour lequel cette borne est atteinte est dit *optimal*.

On obtient pour cette métrique le résultat du théorème 26, comparable à celui du théorème 25. Dans ce théorème $M^t(x,\cdot)$ désigne la loi de probabilité μ sur Ω définie par :

$$\forall y \in \Omega, \quad \mu(y) = M^t(x,y).$$

Théorème 26 (convergence, distance en variation totale)**.** *On considère M une matrice stochastique, associée à une chaîne de Markov sur un ensemble d'états Ω,* ***irréductible*** *et* ***apériodique*** *admettant la distribution stationnaire π, il existe alors une constante réelle $\alpha \in]0,1[$ et un coefficient C strictement positif tels que :*

$$\max_{x\in\Omega} \left\| M^t(x,\cdot) - \pi \right\|_{TV} \leqslant C\alpha^t.$$

Il est clair que le point clef de l'utilisation des méthodes de Monte-Carlo par Chaîne de Markov consiste à obtenir une majoration efficace de $\max_{x\in\Omega} \left\| M^t(x,\cdot) - \pi \right\|_{TV}$.

Pour tout nombre entier naturel t, on note $d(t)$ le maximum de la distance en variation totale entre $M^t(x,\cdot)$ et la distribution stationnaire π :

$$d(t) \stackrel{\text{def}}{=} \max_{x\in\Omega} \left\| M^t(x,\cdot) - \pi \right\|_{TV}.$$

Parmi les techniques classiques qui permettent de majorer $d(t)$ l'une d'entre elles consiste à majorer $\bar{d}(t)$ où :

$$\bar{d}(t) \stackrel{\text{def}}{=} \max_{(x,y)\in\Omega^2} \left\| M^t(x,\cdot) - M^t(y,\cdot) \right\|_{TV}.$$

Il est en effet assez souvent facile de déterminer une majoration efficace de $\bar{d}(t)$ et on peut par ailleurs démontrer que :

$$\forall t \in \mathbb{N}, \quad d(t) \leqslant \bar{d}(t) \leqslant 2d(t).$$

Une conséquence immédiate de ce résultat est que :

$$d(t) \underset{t\to\infty}{=} \Theta\left(\bar{d}(t)\right).$$

Une des commodités de la fonction $\bar{d}$ par rapport à la fonction d tient au fait que $\bar{d}$ est sous-multiplicative :

$$\forall (t,s) \in \mathbb{N}^2, \quad \bar{d}(s+t) \leqslant \bar{d}(s)\cdot\bar{d}(t). \tag{3.4}$$

Temps de mélange

Les notations de la section précédente permettent de définir un paramètre qui mesure le temps (en fait le nombre d'étapes ou le nombre d'itérations du processus) nécessaires pour que la chaîne parviennent à une distance donnée de la distribution stationnaire. On appelle *temps de mélange* et on note t_{mix} ce paramètre.

Dans la pratique t_{mix} désigne à la fois une fonction et une constante. En effet :

- t_{mix} est l'application définie par :

$$t_{\text{mix}} : \begin{array}{ccc}]0,+\infty[& \to & \mathbb{N} \\ \varepsilon & \mapsto & \min\{t\in\mathbb{N}, \quad d(t) \leqslant \varepsilon\}. \end{array}$$

- t_{mix} est aussi la constante définie par :

$$t_{\text{mix}} \overset{\text{def}}{=} t_{\text{mix}}\left(\frac{1}{4}\right).$$

Le choix de $\varepsilon = \frac{1}{4}$ pour définir la constante t_{mix} est lié à l'inégalité 3.4 qui permet d'établir que pour tout nombre réel positif ℓ :

$$d(\ell t_{\text{mix}}) \leqslant 2^{-\ell}.$$

On établit également ainsi un lien très commode entre la fonction et la constante t_{mix} :

$$\forall \varepsilon > 0, \quad t_{\text{mix}}(\varepsilon) \leqslant \left\lceil \log_2\left(\varepsilon^{-1}\right)\right\rceil t_{\text{mix}}. \tag{3.5}$$

Couplage

La notion de couplage de chaînes de Markov s'appuie sur l'égalité 3.3 qui donne une façon possible de définir la distance en variation totale entre deux lois de probabilité.

Formellement un *couplage* de chaînes de Markov est une suite de couples de variables aléatoires $(X_t, Y_t)_{t\in\mathbb{N}}$ telles que $(X_t)_{t\in\mathbb{N}}$ et $(Y_t)_{t\in\mathbb{N}}$ soient deux chaînes de Markov à valeurs dans le même espace d'états Ω associées à la même matrice de transition M.

Tout couplage de chaînes de Markov $(X_t, Y_t)_{t\in\mathbb{N}}$ peut être modifié afin que les deux chaînes coïncident dès qu'elles se rencontrent :

$$\forall s \in \mathbb{N}, \quad (X_s = Y_s) \Rightarrow (\forall t \geqslant s, \quad X_t = Y_t). \tag{3.6}$$

Le résultat du théorème 27 donne un moyen d'obtenir une majoration de t_{mix}.

Théorème 27 (Couplage de chaîne de Markov)**.** *On considère $(X_t, Y_t)_{t\in\mathbb{N}}$ un couplage de chaînes de Markov vérifiant la propriété 3.6, tel que $X_0 = x$ et $Y_0 = y$. On note $\tau_{couplage}$ la première itération pour laquelle les deux chaînes sont égales :*

$$\tau_{couplage} \overset{def}{=} \min\{t \in \mathbb{N}, \quad X_t = Y_t\}.$$

Alors :

$$\left\|M^t(x,\cdot) - M^t(y,\cdot)\right\|_{TV} \leqslant \mathbb{P}_{x,y}\left(\tau_{couplage} > t\right),$$

où $\mathbb{P}_{x,y}$ est la probabilité associée aux événements caractérisés par les points de départ x et y des deux chaînes couplées.

En particulier si pour tout couple $(x, y) \in \Omega^2$ il existe un couplage (X_t, Y_t) tel que $X_0 = x$ et $Y_0 = y$. Pour tout couple (x, y) on note τ_{couplage} la première itération pour laquelle les deux chaînes sont égales, on a :

$$d(t) \leqslant \max_{(x,y)\in\Omega^2} \mathbb{P}_{x,y}\left(\tau_{\text{couplage}} > t\right). \tag{3.7}$$

Dans le cas général, en déterminant un majorant de $\mathbb{P}_{x,y}\left(\tau_{\text{couplage}} > t\right)$ qui ne dépend pas des états de départs x et y, on obtient donc un majorant de $\max_{(x,y)\in\Omega^2} \left\|M^t(x,\cdot) - M^t(y,\cdot)\right\|_{TV}$ donc de $\bar{d}(t)$ et de $d(t)$.

Cas de la marche aléatoire sur un 6-cycle Dans le cas de la marche aléatoire sur le 6-cycle, on obtient un couplage (X_t, Y_t) en considérant deux particules qui se déplacent suivant les chaînes (X_t) et (Y_t) respectivement. On tire au sort de façon équitable laquelle des deux particules bouge et on décide du mouvement de la particule tirée au sort grâce à la matrice de transition M (la version non paresseuse de la marche aléatoire). On procède ainsi jusqu'à ce que les particules soient au même point, leurs mouvements sont ensuite identiques. Il est aisé de constater qu'on obtient bien ainsi un couplage des deux chaînes régies par la même matrice de transition M' (celle qui correspond à la version paresseuse de la marche).

On peut déterminer les positions respectives des deux particules en comptant le nombre de sommets intermédiaires pour passer de l'une à l'autre en tournant dans le sens horaire. Ce nombre de sommets intermédiaires, noté D_t, varie de 0, les deux particules sont sur le même sommet, à 6 (le nombre de sommets intermédiaires à l'étape précédente est 5 et la distance dans le sens horaire augmente de 1) les deux particules sont à nouveau confondues. La variable aléatoire D_t correspond à une marche aléatoire sur les sommets $\{0, 1, 2, 3, 4, 5, 6\}$, où les sommets 0 et 6 sont absorbants. Dans ce cas on a :

$$\tau_{\text{couplage}} = \min\{t \geqslant 0, \quad D_t \in \{0, 6\}\}.$$

Si $D_0 = k$, on démontre que :

$$\mathbb{E}_{x,y}(\tau_{\text{couplage}}) = k(6 - k).$$

Quels que soient les deux points de départ des deux particules, on a donc :

$$\mathbb{E}_{x,y}\left(\tau_{\text{couplage}}\right) \leqslant 9.$$

L'inégalité de Markov assure que :

$$\mathbb{P}_{x,y}\left(\tau_{\text{couplage}} > t\right) \leqslant \frac{\mathbb{E}_{x,y}\left(\tau_{\text{couplage}}\right)}{t},$$

l'inégalité 3.7 permet alors d'établir que :

$$d(t) \leqslant \frac{9}{t}. \tag{3.8}$$

Finalement, pour le 6-cycle, l'inégalité 3.8 assure que le temps de mélange t_{mix} vérifie $t_{\text{mix}} \leqslant 36$. On peut comparer ce résultat à celui obtenu par le théorème 25 en gardant à l'esprit que pour la marche aléatoire paresseuse sur le 6-cycle $\rho = \frac{3}{4}$ et que la majoration proposée par le théorème 25 introduit une constante cachée.

Cas de l'hypercube Le cas de l'hypercube est un peu plus élaboré. L'hypercube de dimension n est le graphe dont les sommets sont les éléments de l'ensemble $\{0, 1\}^n$. Deux sommets sont reliés par une arête si et seulement si ils diffèrent d'une seule coordonnée exactement. La figure 3.6 donne une représentation de l'hypercube de dimension 4.

Les déplacements d'un sommet à un des n sommets adjacents consiste à tirer aléatoirement une des n coordonnées et à lui additionner 1 (modulo 2). En clair chacun des n sommets adjacents est choisi par un tirage aléatoire uniforme. Cette version de la chaîne est périodique puisque tous les cycles sont de longueur paire. La version paresseuse de la marche aléatoire consiste à rester sur le sommet courant avec une probabilité égale à 1/2 et, dans le cas d'un changement de sommet, à choisir de façon aléatoire uniforme un des n sommets adjacents.

Un moyen simple d'obtenir la marche paresseuse sur l'hypercube consiste à d'abord tirer de façon aléatoire et uniforme une des n coordonnées et à affecter à cette coordonnée le résultat d'un tirage de Bernoulli équilibré à valeurs dans $\{0, 1\}$.

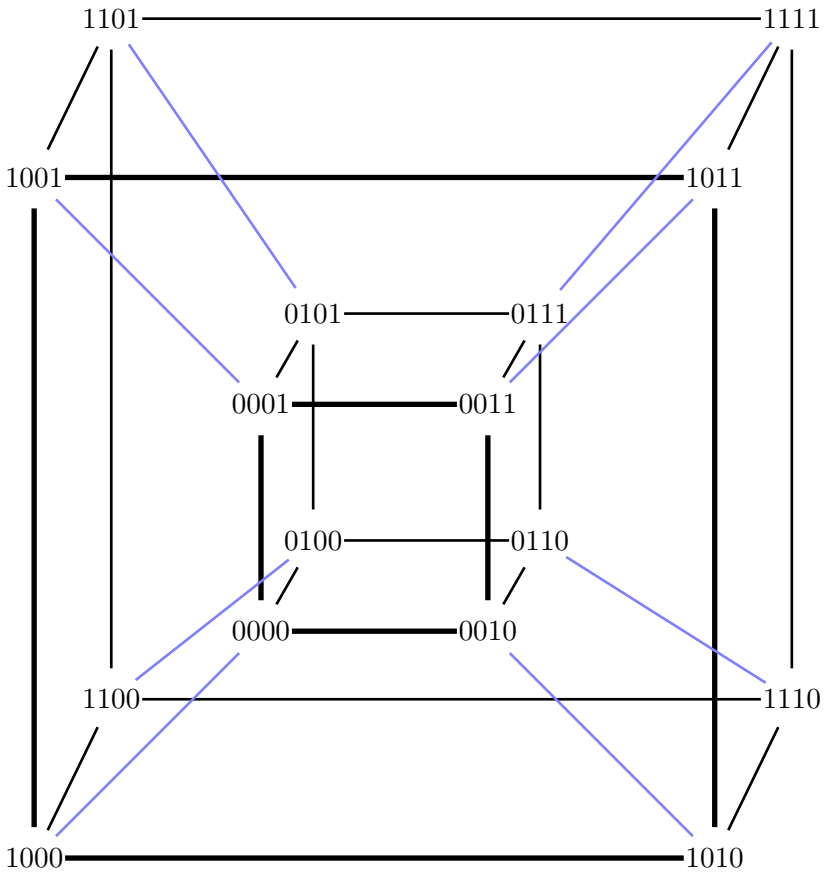


FIGURE 3.6 – Hypercube de dimension 4

Un couplage de deux marches aléatoires commençant à des sommets éventuellement distincts est alors facilement mis en place en appliquant aux deux marches le même procédé qui consiste à tirer une des n coordonnées puis une valeur dans $\{0, 1\}$. Le nombre de coordonnées qui coïncident dans les deux marches aléatoires est une fonction croissante (au sens large) du nombre d'étapes du processus.

En imaginant les 5 tirages suivants dans l'hypercube de dimension 4 pour les deux points de départ $x_0 = 0011$ et $y_0 = 1000$:

$$(2, 1),\ (3, 0),\ (4, 1),\ (3, 1),\ (1, 1),$$

on obtient les déplacements de la figure 3.7. À la fin de ces tirages, les deux chaînes coïncident et suivent alors les mêmes déplacements.

Dès que chacune des n coordonnées aura été sélectionnée par tirage au sort au moins une fois, les deux particules ont des déplacements qui coïncident (leurs déplacements coïncident éventuellement avant que cet événement se réalise). Le temps de couplage τ_{couplage} pour l'hypercube est donc majoré par le nombre de tirages τ nécessaires pour atteindre les n coordonnées au moins une fois.

L'espérance de la variable aléatoire τ est un résultat classique puisqu'il s'agit du temps d'attente moyen du collectionneur de n figurines, c'est-à-dire $n \log(n)$.
Par ailleurs, pour tout nombre réel strictement positif c, en notant A_i l'événement qui correspond au fait que la i-ième coordonnée n'apparaisse pas dans les $\lceil n \log(n) + cn \rceil$ premiers tirages, on a :

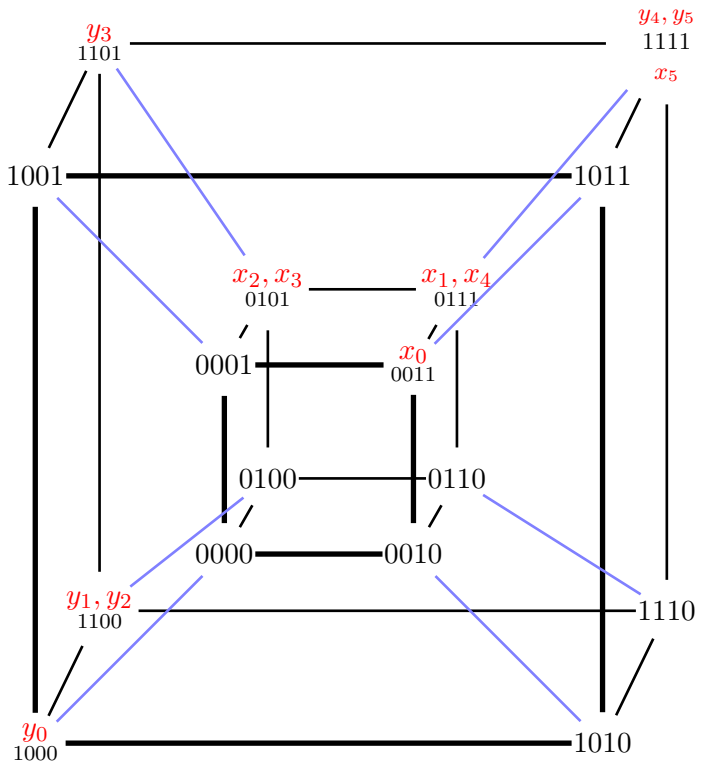


FIGURE 3.7 – Couplage sur l'hypercube de dimension 4

$$\mathbb{P}\left(\tau > \lceil n\log(n) + cn\rceil\right) = \mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right). \tag{3.9}$$

À chaque tirage, la probabilité de ne pas obtenir la i-ième coordonnée est égale à $1-\frac{1}{n}$, on a donc :

$$\forall i \in [\![1,n]\!], \quad \mathbb{P}\left(A_i\right) = \left(1-\frac{1}{n}\right)^{\lceil n\log(n)+cn\rceil}. \tag{3.10}$$

Les formules 3.9 et 3.10 permettent donc d'établir :

$$\begin{aligned}
\mathbb{P}\left(\tau > \lceil n\log(n) + cn\rceil\right) &\leqslant \sum_{i=1}^{n} \mathbb{P}\left(A_i\right) \\
&\leqslant n\left(1-\frac{1}{n}\right)^{\lceil n\log(n)+cn\rceil} \\
&\leqslant \underbrace{n\exp\left(-\frac{n\log(n)+cn}{n}\right)}_{=\exp(-c)}
\end{aligned}$$

.

On a donc :

$$d\left(\lceil n\log(n) + cn\rceil\right) \leqslant \mathbb{P}\left(\tau > \lceil n\log(n) + cn\rceil\right) \leqslant \exp(-c).$$

On obtient donc immédiatement :

$$\boxed{t_{\text{mix}}(\varepsilon) \leqslant n\left(\log(n) + \log\left(\frac{1}{\varepsilon}\right)\right).} \tag{3.11}$$

En particulier :

$$t_{\text{mix}} = \mathcal{O}\left(n \log(n)\right).$$

Ce résultat est important à plus d'un titre. D'abord parce que le problème de la majoration de t_{mix} dans les chaînes de Markov est généralement une question très épineuse. Par ailleurs, ce résultat sera ré-investi dans la section 5.3.2.

3.1.4 Algorithmes d'acceptation-rejet

Algorithmes d'acceptation-rejet, cas général

Cette section présente des résultats bien connus dont les rappels ne sont pas indispensables pour la plupart des lecteurs.

Le problème général auquel s'applique les méthodes d'acceptation-rejet consiste à s'interroger sur la façon de simuler une variable aléatoire X dont la fonction de répartition est F :

$$F(x) = \mathbb{P}(X \leqslant x).$$

On suppose que F admet une fonction de densité f. On se place donc dans une hypothèse générale de probabilités continues qui englobera le cas discret que nous évoquerons aussi.

On suppose également que l'on dispose d'une fonction de répartition H, associée à une densité h, pour laquelle on dispose déjà d'un générateur efficace.

On suppose enfin que le rapport (positif) $\dfrac{f(x)}{h(x)}$ est majoré par une constante réelle strictement positive c.

La méthode d'acception-rejet se présente alors de la façon suivante :

Pour engendrer une variable aléatoire Y admettant la distribution F :

1. On engendre une variable aléatoire Z distribuée selon H. C'est-à-dire que l'on tire une valeur z de Z suivant la loi définie par H.
2. On engendre, indépendamment de Z une variable aléatoire U à valeurs dans $[0, 1]$, suivant la loi uniforme. On tire donc aléatoirement et uniformément dans $[0, 1]$ une valeur $u \in [0, 1]$.
3. Si $u \leqslant \dfrac{f(z)}{c \cdot h(z)}$, alors Y prend la valeur z (*acceptation*).
   Sinon on retourne à l'étape 1 (*rejet*).

Le fonctionnement de la méthode s'appuie en partie sur les remarques préliminaires suivantes :

- Les variables aléatoires $f(Z)$, $h(Z)$ et $\dfrac{f(Z)}{c \cdot h(Z)}$ sont indépendantes de la variable aléatoire U qui assure le tirage uniforme dans $[0, 1]$.
- En supposant les fonctions f et h à valeurs strictement positives, les hypothèses faites à propos de f et h assurent que le rapport $\dfrac{f(Z)}{c \cdot h(Z)}$ est à valeurs dans $]0, 1]$.
- Le nombre N d'itérations des étapes 1 et 2 de la méthode est elle même une variable aléatoire qui suit une loi géométrique de paramètre p défini par :

  $$p \overset{\text{def}}{=} \mathbb{P}\left(U \leqslant \frac{f(Z)}{c \cdot h(Z)}\right).$$

  Pour tout nombre entier naturel non-nul n, on a donc $\mathbb{P}\left(N = n\right) = (1-p)^{n-1}p$ et l'espérance de N vérifie $\mathbb{E}(N) = \frac{1}{p}$.
- Lorsque l'étape 3 est réalisée, la variable aléatoire Y obtenue à la distribution de Z conditionnellement à l'événement $\left(U \leqslant \frac{f(Z)}{c \cdot h(Z)}\right)$.

Il est alors intéressant de noter que :

$$\mathbb{P}\left(U \leqslant \frac{f(Z)}{c \cdot h(Z)} \middle| Z = z\right) = \frac{f(z)}{c \cdot h(z)}$$

et le paramètre p vérifie donc :

$$p = \int_{\mathbb{R}} \frac{f(z)}{c \cdot h(z)} \cdot h(z) \mathrm{d}z = \frac{1}{c} \underbrace{\int_{\mathbb{R}} f(z)\, \mathrm{d}z}_{=1} = \frac{1}{c}.$$

Le nombre moyen de rejets étant égal à c, on a donc intérêt à l'optimiser en étant au plus près de

$$\sup_{\mathbb{R}} \frac{f(x)}{h(x)}.$$

Ces remarques ne suffisent pas à valider le fait que la méthode fonctionne, c'est-à-dire que la distribution conditionnelle de Z sachant que $U \leqslant \frac{f(Z)}{c \cdot h(Z)}$ est effectivement F.
Pour parvenir à ce résultat, on pose $A \stackrel{\text{def}}{=} (Z \leqslant z)$ et $B \stackrel{\text{def}}{=} \left(U \leqslant \frac{f(Z)}{c \cdot h(Z)}\right)$.
La formule de Bayes : $\mathbb{P}(A|B) \cdot \mathbb{P}(B) = \mathbb{P}(B|A) \cdot \mathbb{P}(A)$ s'écrit ici :

$$\mathbb{P}\left(Z \leqslant z \middle| U \leqslant \frac{f(Z)}{c \cdot h(Z)}\right) \cdot \underbrace{\mathbb{P}\left(U \leqslant \frac{f(Z)}{c \cdot h(Z)}\right)}_{=p=\frac{1}{c}} = \mathbb{P}\left(U \leqslant \frac{f(Z)}{c \cdot h(Z)} \middle| Z \leqslant z\right) \cdot \underbrace{\mathbb{P}\left(Z \leqslant z\right)}_{=H(z)}.$$

Par ailleurs :

$$\begin{aligned}
\mathbb{P}\left(U \leqslant \frac{f(Z)}{c \cdot h(Z)} \middle| Z \leqslant z\right) &= \frac{\mathbb{P}\left(U \leqslant \frac{f(Z)}{c \cdot h(Z)} \bigcap Z \leqslant z\right)}{\mathbb{P}\left(Z \leqslant z\right)} \\
&= \frac{\mathbb{P}\left(U \leqslant \frac{f(Z)}{c \cdot h(Z)} \bigcap Z \leqslant z\right)}{H\left(z\right)} \\
&= \int_{-\infty}^{z} \frac{\mathbb{P}\left(U \leqslant \frac{f(\zeta)}{c \cdot h(\zeta)}\right)}{H\left(z\right)}\, h(\zeta)\, \mathrm{d}\zeta \\
&= \frac{1}{H(z)} \int_{-\infty}^{z} \frac{f(\zeta)}{c \cdot h(\zeta)} h(\zeta)\, \mathrm{d}\zeta \\
&= \frac{1}{cH(z)} \underbrace{\int_{-\infty}^{z} f(\zeta)\, \mathrm{d}\zeta}_{=F(z)}.
\end{aligned}$$

On a donc finalement :

$$\mathbb{P}\left(Z \leqslant z \middle| U \leqslant \frac{f(Z)}{c \cdot h(Z)}\right) = c \frac{F(z)}{cH(z)} H(z) = F(z).$$

Ce qui valide la méthode.

Algorithmes d'acceptation-rejet, cas discret

Le cas discret est analogue au cas continu, on souhaite simuler une variable aléatoire Y à valeurs dans une partie de $\mathbb{N}$ et telle que pour pour tout $k \in \mathbb{N}$, $\mathbb{P}(Y = k) = p(k)$. On dispose d'un générateur pour simuler une variable aléatoire discrète Z pour laquelle $\mathbb{P}(Z = k) = q(k)$. On suppose par ailleurs qu'il existe c un nombre réel strictement positif tel que pour tout nombre entier naturel k, $\frac{p(k)}{q(k)} \leqslant c$. L'algorithme se présente de la façon suivante :

1. On engendre une variable aléatoire Z qui suit la distribution q. En clair on tire k avec la probabilité $q(k)$.
2. On engendre indépendamment de Z, une variable aléatoire U à valeurs dans $[0,1]$, qui suit la loi uniforme. On tire donc de façon aléatoire et uniforme $u \in [0,1]$.
3. Si $u \leqslant \frac{p(k)}{cq(k)}$, la variable aléatoire Y prend la valeur k (*acceptation*), on retourne à l'étape 1 sinon (*rejet*).

3.1.5 L'algorithme de Metropolis-Hastings

Ici, on suppose que le générateur est construit à partir d'une méthode de Monte-Carlo par chaîne de Markov. On suppose que la chaîne est associée à une matrice de transition M sur un espace d'états Ω. Le but de l'algorithme est de modifier la chaîne afin d'obtenir une distribution stationnaire π arbitraire.

L'idée telle qu'elle est exposée dans [CG95] consiste à modifier M afin que si $M(x,\cdot)$ propose un déplacement en y, alors ce déplacement est accepté avec une probabilité $a(x,y)$ et rejeté avec la probabilité $1-a(x,y)$. En cas de rejet, on "boucle" donc sur l'état x. La méthode se distingue donc ici des algorithmes d'acceptation-rejet exposés dans la section précédente. En effet pour ces algorithmes, en cas de rejet, on effectue un nouveau tirage (en fait deux) indépendant(s) du précédent.

On obtient la nouvelle matrice de transition P définie par :

$$\forall (x,y) \in \Omega^2, \quad P(x,y) \stackrel{\text{def}}{=} \begin{cases} M(x,y)a(x,y) & \text{si } x \neq y \\ 1 - \displaystyle\sum_{z \in \Omega \setminus \{x\}} M(x,z)a(x,z) & \text{si } x = y \end{cases}. \tag{3.12}$$

Il s'agit alors de définir $a(x,y)$, dans la formule 3.12, pour que l'objectif soit atteint de la façon la plus efficace. Il faut d'une part atteindre la distribution cible π et optimiser $a(x,y)$ afin de ne pas trop "ralentir" les déplacements dans la chaîne par des rejets trop fréquents.

La distribution stationnaire π est atteinte si :

$$\forall (x,y) \in \Omega^2, \quad \pi(x)M(x,y)a(x,y) = \pi(y)M(y,x)a(y,x). \tag{3.13}$$

Trois situation peuvent se produire :

1. $\pi(x)M(x,y) = \pi(y)M(y,x)$
2. $\pi(x)M(x,y) > \pi(y)M(y,x)$
3. $\pi(x)M(x,y) < \pi(y)M(y,x)$

Dans le premier cas, il suffit de poser $a(x,y) \stackrel{\text{def}}{=} 1$, $a(y,x) \stackrel{\text{def}}{=} 1$ pour que 3.13 soit vérifiée.

Le second cas correspond à une situation que l'on pourrait décrire informellement en disant que le processus assure des passages de x à y trop "fréquents" par rapport à des passages de y à x trop "rares". Dans cette configuration on pose $a(y,x) \stackrel{\text{def}}{=} 1$ et, puisque $\pi(x)M(x,y)$ est nécessairement strictement positif, on peut définir $a(x,y)$ par :

$$a(x,y) \stackrel{\text{def}}{=} \frac{\pi(y)M(y,x)}{\pi(x)M(x,y)}.$$

Par construction on a alors $a(x,y) \in [0,1[$.

Il est clair que le troisième cas est exactement symétrique du deuxième.

On peut résumer tous les cas en définissant $a(x,y)$ par :

$$a(x,y) \stackrel{\text{def}}{=} \begin{cases} \min\left\{1, \frac{\pi(y)M(y,x)}{\pi(x)M(x,y)}\right\} & \text{si } \pi(x)M(x,y) > 0 \\ 1 & \text{sinon} \end{cases}, \tag{3.14}$$

et $a(x,y) = 0$ si $M(x,y) = 0$.

En notant $\left[1 \wedge \frac{\pi(y)M(y,x)}{\pi(x)M(x,y)}\right]$ le nombre réel $a(x,y)$ défini par la formule 3.14, l'équation 3.12 s'écrit :

$$\forall (x,y) \in \Omega^2, \quad P(x,y) \stackrel{\text{def}}{=} \begin{cases} M(x,y)\left[1 \wedge \frac{\pi(y)M(y,x)}{\pi(x)M(x,y)}\right] & \text{si } x \neq y \\ 1 - \sum_{z \in \Omega \setminus \{x\}} M(x,z)\left[1 \wedge \frac{\pi(z)M(z,x)}{\pi(x)M(x,z)}\right] & \text{si } x = y \end{cases}. \tag{3.15}$$

Il est d'abord clair, qu'ainsi définie, la matrice P est à coefficients positifs, par ailleurs pour tout $x \in \Omega$, on a :

$$\begin{aligned} \sum_{y \in \Omega} P(x,y) &= P(x,x) + \sum_{y \in \Omega \setminus \{x\}} P(x,y) \\ &= 1 - \sum_{z \in \Omega \setminus \{x\}} M(x,z)\left[1 \wedge \frac{\pi(z)M(z,x)}{\pi(x)M(x,z)}\right] + \sum_{y \in \Omega \setminus \{x\}} M(x,y)\left[1 \wedge \frac{\pi(y)M(y,x)}{\pi(x)M(x,y)}\right] \\ &= 1. \end{aligned}$$

La matrice P est donc stochastique. Par ailleurs, par construction P vérifie l'équation de réversibilité temporelle 3.13. L'objectif est donc atteint.

Il est important de noter que si M est symétrique, la définition 3.15 de la matrice de transition P devient :

$$\forall (x,y) \in \Omega^2, \quad P(x,y) \stackrel{\text{def}}{=} \begin{cases} M(x,y)\left[1 \wedge \frac{\pi(y)}{\pi(x)}\right] & \text{si } x \neq y \\ 1 - \sum_{z \in \Omega \setminus \{x\}} M(x,z)\left[1 \wedge \frac{\pi(z)}{\pi(x)}\right] & \text{si } x = y \end{cases}. \tag{3.16}$$

On peut remarquer que dans les deux cas (3.15 et 3.16) c'est le rapport $\frac{\pi(y)}{\pi(x)}$ qui guide les transitions de la nouvelle chaîne. Par ailleurs, dans le cas où M est symétrique, lorsque la chaîne initiale (dirigée par M) propose une transition de x à y (avec $x \neq y$) celle-ci est toujours acceptée dans le cas où $\pi(y) \leqslant \pi(x)$, elle est acceptée avec une probabilité non-nulle mais strictement inférieure à 1 dans le cas contraire.

Dans tous les cas, que la matrice M soit symétrique ou non, le fonctionnement du générateur aléatoire s'appuyant sur l'algorithme de Metropolis-Hastings pour engendrer N éléments est le suivant :

1. On choisit une valeur arbitraire initiale x_0,
2. Pour j variant de 0 à N, on tire y parmi les voisins de x en s'appuyant sur la distribution $M(x_j, \cdot)$,
3. On tire indépendamment et uniformément une valeur de $u \in [0,1]$,
4. Si $u \leqslant a(x_j, y)$, on pose $x_{j+1} \stackrel{\text{def}}{=} y$,

5. Si $u > a(x_j, y)$, on pose $x_{j+1} \stackrel{\text{def}}{=} x_j$,
6. On renvoie la suite $x_0, x_1, \ldots, x_N$.

Les suites ainsi produites simulent les déplacements dans un graphe associé à une chaîne de Markov dont la distribution stationnaire est égale à π.

Il est clair que la méthode ne règle rien des soucis de vitesse de convergence. Les questions qui restent à régler dans cette affaire sont liées aux temps de mélange dans la chaîne.

3.2 Quelques principes de statistiques

Dans le chapitre 6 on montre comment utiliser des tests statistiques classiques pour valider une hypothèse sur le temps de mélange dans un générateur qui s'appuie sur un algorithme de Monte-Carlo par chaîne de Markov (MCMC).

Dans ce qui suit, sont faits quelques rappels de statistiques. On suppose néanmoins le lecteur familier avec quelques notions de base. En particulier les notions d'espace probabilisé, de variable aléatoire (toutes supposées à valeurs réelles) sont supposées connues. Le lecteur intéressé pourra se référer à [Lej04] et [Riv12] qui ont en partie inspiré les notes qui suivent.

Deux tests spécifiques aux algorithmes MCMC sont également évoqués. Il s'agit des tests de Gelman Rubin et d'autocorrélation.

Le premier permet d'évaluer l'influence du point de départ dans le graphe associé à la chaîne de Markov. Le second test est destiné à mesurer la corrélation entre des valeurs engendrées à une même distance fixée.

3.2.1 Principes généraux utiles de probabilité-statistique

Dans le titre de cette section, l'adjectif *utile* ne se réfère qu'aux résultats qui seront utilisés dans ce mémoires. Il est bien clair qu'il ne s'agit pas ici de faire un cours de statistiques. Néanmoins l'exposé s'efforcera d'établir et de faire saisir les idées importantes.

L'objet de la statistique et les questions auxquelles elle s'attache sont liés à l'étude d'observations répétées issues d'un certain phénomène aléatoire. Classiquement, l'aléa peut être expérimental, quand il est le résultat par exemple d'un sondage d'une population ou lié à une modélisation lorsqu'il s'agit par exemple de choisir une loi de probabilité qui permette de théoriser la durée de vie d'un appareil.

Dans notre étude la loi de probabilité est connue, c'est la distribution stationnaire π associée à une chaîne de Markov. Les outils statistiques vont nous servir en fait à tester une hypothèse de convergence. Le nombre de pas effectués dans le graphe associé à la chaîne peut-il être jugé suffisant pour valider le générateur comme simulateur acceptable de la loi π ?

Fonction de répartition, densité de probabilité, fonction génératrice des moments, lois usuelles utiles

Fonction de répartition, densité de probabilité Pour une variable aléatoire X, on appelle **fonction de répartition** de X et on note F_X l'application définie sur $\mathbb{R}$:

$$\forall x \in \mathbb{R}, \quad F_X(x) = \mathbb{P}\left(X \leqslant x\right).$$

La notion de fonction de répartition n'est pas indispensable en probabilités discrètes, elle reste néanmoins un outil très pratique. De la même façon la notion de densité de probabilité ne sera finalement utile que pour comprendre le théorème central limite qui est lui même la clef du test du χ^2.

On dit d'une variable aléatoire X qu'elle admet une densité s'il existe une fonction positive f_X intégrable telle que pour tout $x \in \mathbb{R}$:

$$F_X(x) = \int_{-\infty}^{x} f_X(u)\,\mathrm{d}u.$$

Nous n'évoquerons pas ici ni dans la suite les détails théoriques intéressants liés au sens de *fonction intégrable* dans la définition précédente.

On appelle **fonction génératrice des moments** de la variable aléatoire X, l'application notée Ψ_X, définie, lorsque l'expression a du sens par :

$$\forall t \in \mathbb{R}, \quad \Psi_X(t) = \mathbb{E}\left(\exp(tX)\right).$$

Il faut bien distinguer cette fonction de la **fonction génératrice** G_X définie par :

$$\forall t \in \mathbb{R}, \quad G_X(t) = \mathbb{E}\left(t^X\right).$$

Nous évoquerons dans ce travail les lois suivantes :

La loi multinomiale Les lois multinomiales généralisent les lois de Bernoulli à deux issues (succès ou échec), il s'agit simplement de proposer c catégories possibles d'issues possibles de probabilités $p_1, \ldots, p_c$ telles que $\sum_{k=1}^{c} p_k = 1$. On s'intéresse aux variables aléatoires $X_1, \ldots, X_c$ qui comptent le nombres de tirages correspondant à chacune des catégories. On établit que pour tout c-uplet de nombres entiers naturels $(n_1, \ldots, n_c)$ et tout nombre entier naturel n qui compte le nombre d'expériences aléatoires réalisées :

$$\mathbb{P}\left(X_1 = n_1, \ldots, X_c = n_c\right) = \binom{n}{n_1, \ldots, n_c} p_1^{n_1} \ldots p_c^{n_c},$$

où :

$$\binom{n}{n_1, \ldots, n_c} = \begin{cases} \dfrac{n!}{n_1! \ldots n_2!} & \text{si} \quad n_1 + \cdots + n_c = n \\ 0 \quad \text{sinon} \end{cases}.$$

Loi normale Contrairement à la loi précédente, il s'agit d'une loi continue, appelée aussi loi de Gauss. Elle est fondamentale en particulier pour le rôle qu'elle joue dans le théorème central limite 31. Pour tous nombres réels μ et σ on dit que la variable aléatoire X suit *une loi de Gauss* ou *loi normale* notée $\mathcal{N}(\mu, \sigma)$ si elle admet pour densité l'application f définie sur $\mathbb{R}$ par :

$$\forall x \in \mathbb{R}, \quad f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Il faut retenir que dans ce cas, l'espérance et la variance de X sont respectivement égales à μ et σ^2.

Par ailleurs, toute fonction linéaire d'une variable aléatoire gaussienne est gaussienne et toute combinaison linéaire de variables gaussiennes indépendantes est une variable aléatoire gaussienne. On peut retenir plus précisément que :

- Si $Z \rightsquigarrow \mathcal{N}(0,1)$ et si $X \overset{\text{def}}{=} \sigma Z + \mu$ alors $X \rightsquigarrow \mathcal{N}(\mu, \sigma^2)$.
- Si X_1 et X_2 sont deux variables aléatoires indépendantes telles que $X_1 \rightsquigarrow \mathcal{N}(\mu_1, \sigma_1^2)$ et $X_2 \rightsquigarrow \mathcal{N}(\mu_2, \sigma_2^2)$ alors $X_1 + X_2 \rightsquigarrow \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$.

Loi Gamma Comme la loi précédente, il s'agit d'une loi continue qui dépend de deux paramètres. Pour tous nombres réels strictement positifs a et λ on dit que la variable aléatoire X suit *une loi Gamma* notée $\Gamma(a,\lambda)$ si elle admet pour densité l'application g définie sur $\mathbb{R}$ par :

$$\forall x \in \mathbb{R}, \quad g(x) = \frac{\lambda^a}{\Gamma(a)} x^{a-1} \exp(-\lambda x) \cdot \mathbb{1}_{[0,+\infty[}(x),$$

où :

$$\forall a \in]0,+\infty[, \quad \Gamma(a) = \int_0^{+\infty} t^{a-1} \exp(-t)\, \mathrm{d}t.$$

On rappelle que pour tout nombre entier naturel non-nul n, on a :

$$\Gamma(n) = (n-1)!.$$

Il faut retenir que dans ce cas, l'espérance et la variance de X sont respectivement égales à $\frac{a}{\lambda}$ et $\frac{a}{\lambda^2}$.

Par ailleurs, si $X_1, \ldots, X_n$ sont des variables aléatoires indépendantes telles que pour tout $i \in [\![1,n]\!]$, $X_i \rightsquigarrow \Gamma(a_i,\lambda)$ alors :

$$\sum_{i=1}^{n} X_i \rightsquigarrow \Gamma\left(\sum_{i=1}^{n} a_i, \lambda\right).$$

Convergences On rappelle ici brièvement quelques modes essentiels de convergence de la théorie des probabilités.

On considère une suite $(X_n)_{n\in\mathbb{N}}$ une suite de variables aléatoires, X une variable aléatoire. On dit que :

- $(X_n)_{n\in\mathbb{N}}$ **converge en loi** vers X si en tout point x où la fonction de répartition F_X est continue, on a :
$$\lim_{n\to\infty} F_{X_n}(x) = F_X(x).$$
Si c'est le cas on note $X_n \xrightarrow[n\to\infty]{\mathcal{L}} X$.
- $(X_n)_{n\in\mathbb{N}}$ **converge en probabilité** ou **converge faiblement** vers X si pour tout $\varepsilon \in]0,\infty[$:
$$\lim_{n\to\infty} \mathbb{P}(|X_n - X| < \varepsilon) = 1.$$
Si c'est le cas, on note $X_n \xrightarrow[n\to\infty]{\mathrm{p}} X$.
- $(X_n)_{n\in\mathbb{N}}$ **converge presque sûrement** ou **converge fortement** vers X si pour tout $\varepsilon \in]0,\infty[$:
$$\lim_{n\to\infty} \mathbb{P}\left(\sup_{m\geqslant n} (|X_m - X| < \varepsilon)\right) = 1.$$
Si c'est le cas, on note $X_n \xrightarrow[n\to\infty]{\mathrm{p.s.}} X$.

La hiérarchie est la suivante : la convergence presque sûrement implique la convergence en probabilité qui elle même implique la convergence en loi.

Loi des grands nombres Nous présentons ici la loi forte des grands nombres qui donne un résultat assez intuitif.

Théorème 28 (loi forte des grands nombres)**.** *On considère une suite $(X_n)_{n\in\mathbb{N}}$ une suite de variables aléatoires indépendantes et de même loi d'espérance μ, alors :*

$$\frac{1}{n}\sum_{k=1}^{n} X_k \xrightarrow[n\to\infty]{p.s.} \mu.$$

Ce théorème assure que la moyenne empirique de variables aléatoires indépendantes de même loi, tend presque sûrement vers l'espérance de la loi. En revanche, le théorème ne dit rien du mode de convergence.

Échantillon Le point clef des statistiques est lié à la constitution d'échantillons.

Définition 29 (n-échantillon)**.** *On considère n un nombre entier naturel non-nul, ν une loi de probabilité, on appelle* ***échantillon aléatoire de taille*** *n, ou tout simplement n-échantillon, toute famille de n variables aléatoires indépendantes identiquement distribuées selon la même loi ν.*
On dit parfois que ν est la ***loi mère*** *de l'échantillon.*

Un n-échantillon se présente donc sous la forme d'une famille de n variables aléatoires $(X_1, \ldots, X_n)$ pouvant prendre par exemple la valeur $(x_1, \ldots, x_n)$. On parle parfois en statistique, à propos de $(x_1, \ldots, x_n)$ de *l'échantillon réalisé* qui correspond aux valeurs observées suite à une expérience aléatoire.

Définition 30 (statistique)**.** *On considère $(X_1, \ldots, X_n)$ un n-échantillon, on appelle* ***statistique*** *toute variable aléatoire S_n de la forme $h(X_1, \ldots, X_n)$.*

Le théorème central limite Le théorème central limite précise par rapport à la loi des grands nombres la façon dont la moyenne empirique converge vers l'espérance de la *loi mère.*

Théorème 31 (théorème central limite)**.** *On considère $(X_n)_{n\in\mathbb{N}}$ une suite de variables aléatoires indépendantes et de même loi d'espérance μ et de variance σ^2, alors la suite $\left(\frac{\sqrt{n}}{\sigma}\left(\frac{1}{n}\sum_{k=1}^{n} X_k - \mu\right)\right)_{n\in\mathbb{N}}$ converge en loi vers la loi normalisée centrée réduite :*

$$\frac{\sqrt{n}}{\sigma}\left(\frac{1}{n}\sum_{k=1}^{n} X_k - \mu\right) \xrightarrow[n\to\infty]{\mathcal{L}} \mathcal{N}(0,1)$$

L'apport essentiel du théorème central limite consiste à obtenir une convergence en loi vers une gaussienne quelle que soit la loi mère des variables aléatoires X_i.

la loi du χ^2

Il existe beaucoup de distributions fondamentales en théorie statistique. On a vu le caractère essentiel de la loi normale $\mathcal{N}(\mu, \sigma^2)$ et le rôle que cette loi joue dans le théorème central limite. On expose dans cette section une autre loi intimement liée aux lois normales, indispensable pour comprendre le test du χ^2.

Définition 32 (loi du χ_2)**.** *On considère $Z_1, \ldots, Z_d$ une suite de variables aléatoires indépendantes identiquement distribuées de loi $\mathcal{N}(0,1)$. La loi suivie par la variable aléatoire $\sum_{i=1}^{d} Z_i^2$ est appelée* ***loi du Khi-deux*** *à d degrés de liberté. On la note $\chi^2(d)$.*

Proposition 33 (densité de la loi $\chi^2(d)$)**.** *La densité de la loi du Khi-deux à d degrés de liberté, φ_d est définie par :*

$$\forall x \in \mathbb{R}, \quad \varphi_d(x) = \begin{cases} \dfrac{1}{2^{\frac{d}{2}}\Gamma\left(\frac{d}{2}\right)} x^{\frac{d}{2}-1}\mathrm{e}^{-\frac{x}{2}} & \text{si } x > 0 \\ 0 & \text{sinon} \end{cases}.$$

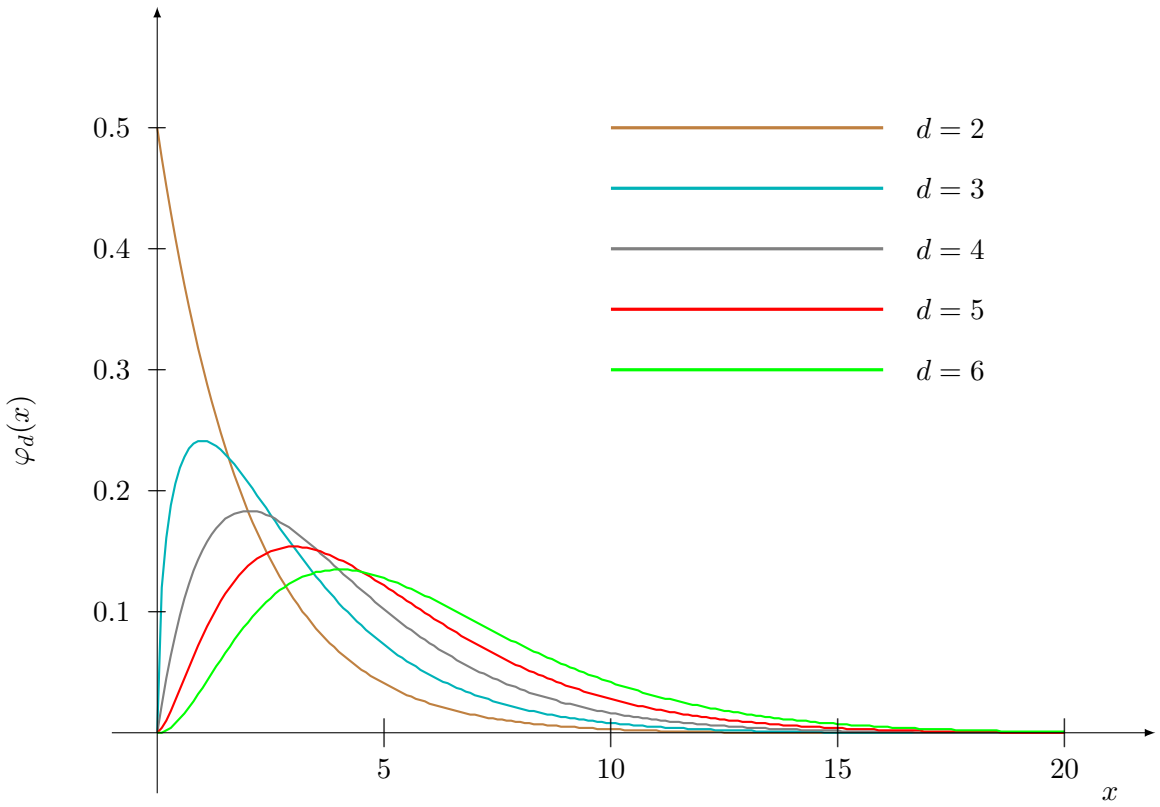


FIGURE 3.8 – Différentes distributions φ_d de la loi du χ^2 pour différentes valeurs du paramètre d

3.2.2 Tests d'hypothèses, le test du χ^2

H_0 contre H_1

Un test statistique a pour but de décider, sur la base d'un échantillon, si une caractéristique de la population vérifie ou non une certaine hypothèse. Cette hypothèse peut porter sur un paramètre θ à valeurs dans $\mathbb{R}$ ou $\mathbb{R}^k$. Plus précisément, le test statistique propose un protocole qui permet d'accepter ou de rejeter une hypothèse du type $\theta \in \Theta_0$.

Cette hypothèse de référence est appelée hypothèse nulle et est notée H_0.

L'hypothèse alternative est notée H_1 et correspond donc à $\theta \in \overline{\Theta}_0$, où $\overline{\Theta}_0$ désigne l'ensemble complémentaire de Θ_0.

Un test pour H_0 est un protocole de décision fondé sur la valeur t d'une statistique T réalisée sur un échantillon. T est appelée la *statistique de test*. Le protocole s'appuie sur la définition d'une partie A de $\mathbb{R}$:

- Si $t \in A$, on accepte H_0.
- Si $t \in \overline{A}$, on rejette H_0.

La région est donc appelée *région d'acceptation* tandis que $\overline{A}$ est appelée *région de rejet*. Il faut bien comprendre que ce protocole est arbitraire et recèle intrinsèquement deux types d'erreur :

- Le rejet de H_0 alors que celle-ci est vraie. On parle alors d'erreur de *première espèce*.
- L'acceptation de H_0 alors que l'hypothèse est fausse. On parle alors d'erreur de *seconde espèce*.

On caractérise chaque erreur par sa probabilité. Dans le cadre de la théorie des tests de décision, les probabilités d'erreur sont appelées des *risques*.

On appelle *risque de première espèce*, et on note souvent α, la probabilité de rejeter H_0 alors que l'hypothèse est vraie.

Par ailleurs, on appelle *puissance d'un test* le probabilité de rejeter H_0 alors que l'hypothèse est effectivement fausse.

On dit qu'un test est *sans biais* lorsque sa puissance est supérieure ou égale à son risque α, la probabilité de rejeter H_0 alors que l'hypothèse est vraie.

p-valeurs

L'art du statisticien consiste à faire un choix au sujet du risque de première espèce α. Pour rendre compte du résultat d'un test la p-valeur est un outil supplémentaire. La p-valeur est en effet la probabilité que sous l'hypothèse H_0, la statistique de test prenne une valeur au moins aussi élevée que celle qui a été observée.

Pour le test du χ^2, la figure 3.9 illustre de façon concrète cette notion.

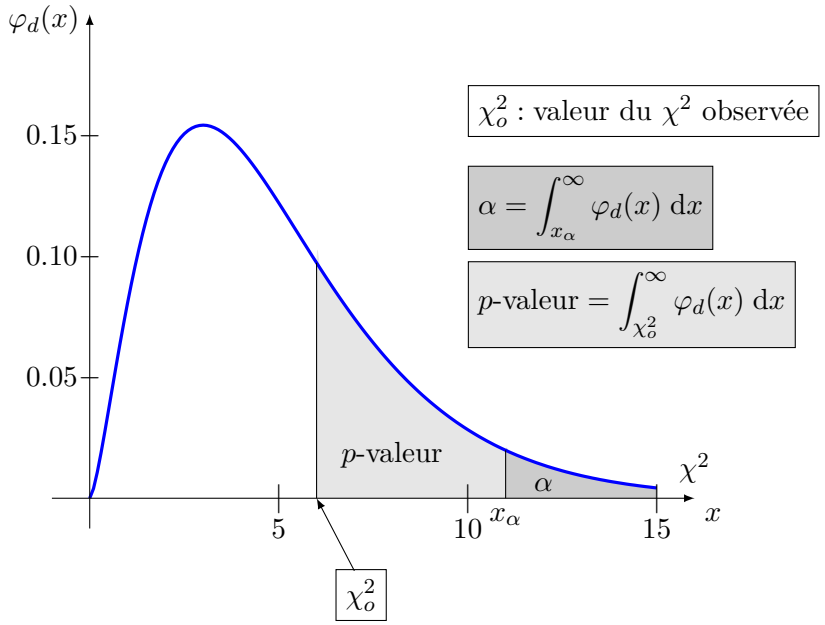


FIGURE 3.9 – x_α est fixé afin d'obtenir la valeur de α souhaitée. La p-valeur est la probabilité d'obtenir une valeur au moins égale à celle qui a été observée : p-valeur $= \int_{\chi^2_o}^{\infty} \varphi_d(x)\,\mathrm{d}x$

Le test du χ^2 s'appuie sur une hypothèse qui n'est pas quantitative, même si la statistique de test l'est. On le classe pour cette raison dans la famille des tests non paramétriques. Puisque c'est ce test que l'on utilise dans la section 6.3.3 on développe un peu plus son principe dans la section suivante.

Le test du χ^2

Il faut en fait parler **des** tests du χ^2, puisque ces tests statistiques présentent plusieurs variantes qui s'adaptent à des natures de tests différentes. Essentiellement, il s'agit de tester :

- un ajustement à une loi simple,
- un ajustement à une famille de lois,
- une propriété d'indépendance,
- une propriété d'homogénéité.

Les tests du χ^2 s'appliquent à des observations discrètes et s'appuient sur le théorème central limite dans sa version vectorielle.

Nous ne nous intéresserons ici uniquement au test d'ajustement à une loi donnée. On considère donc un n-échantillon $X = (X_1, \ldots, X_n)$ constitué de variables aléatoires prenant leurs valeurs dans un ensemble discret $\mathcal{X} = \{x_1, \ldots, x_d\}$ admettant d valeurs

distinctes. Dans la section 6.3.3, il s'agira par exemple des variables aléatoires définies sur une classe d'automates partiellement ordonnés et qui à chaque automate partiellement ordonnés associent le nombre de boucles de cet automate. On note π la loi $(\pi_1, \ldots, \pi_d)$ commune à chacune des variables X_t :

$$\forall t \in [\![1, n]\!], \quad \forall i \in [\![1, n]\!], \quad \mathbb{P}(X_t = x_i) = \pi_i$$

On souhaite réaliser un test des hypothèses :

$$H_0 : \pi = \pi^{\text{ref}} \quad \text{vs} \quad H_1 : \pi \neq \pi^{\text{ref}},$$

où π^{ref} est une loi de référence de support plein sur $\mathcal{X}$, c'est-à-dire que :

$$\sum_{i=1}^{d} \pi_i^{\text{ref}} = 1.$$

En particulier :

$$H_0 : \forall i \in [\![1, n]\!], \quad \pi_i = \pi_i^{\text{ref}},$$

et :

$$H_1 : \exists i \in [\![1, n]\!], \quad \pi_i \neq \pi_i^{\text{ref}}.$$

Pour estimer π, on utilise la méthode des moments. Pour tout $i \in [\![1, n]\!]$:

$$\hat{\pi}_{i,n} = \frac{1}{n} \sum_{t=1}^{n} \mathbb{1}_{\{X_t = x_i\}},$$

est un estimateur *fortement consistant et sans biais* de π_i. La loi forte des grands nombres assure en effet que $\hat{\pi}_{i,n}$ converge fortement vers la fréquence π_i de la valeur x_i (l'estimateur est fortement consistant). Il est *sans biais* dans la mesure ou $\mathbb{E}(\hat{\pi}_{i,n}) = \pi_i$. En effet :

$$\mathbb{E}(\hat{\pi}_{i,n}) = \mathbb{E}\left(\frac{1}{n} \sum_{t=1}^{n} \mathbb{1}_{\{X_t = x_i\}}\right) = \frac{1}{n} \mathbb{E}\left(\sum_{t=1}^{n} \mathbb{1}_{\{X_t = x_i\}}\right) = \frac{1}{n} \sum_{t=1}^{n} \pi_i = \pi_i.$$

Il est donc naturel de construire une statistique de test qui mesure l'écart entre $\hat{\pi}_n$ et π^{ref}. Le bon outil de mesure est la statistique de Pearson, appelée parfois *pseudo-distance du* χ^2, définie par :

$$D_n^2\left(\hat{\pi}_n, \pi^{\text{ref}}\right) \overset{\text{def}}{=} n \sum_{i=1}^{d} \frac{\left(\hat{\pi}_{i,n} - \pi_i^{\text{ref}}\right)^2}{\pi_i^{\text{ref}}} = \frac{1}{n} \sum_{i=1}^{d} \frac{\left(N_{i,n} - n\pi_i^{\text{ref}}\right)^2}{\pi_i^{\text{ref}}}, \tag{3.17}$$

où :

$$N_{i,n} \overset{\text{def}}{=} \mathbb{1}_{\{X_t = x_i\}}.$$

Le théorème 34 donnent les résultats qui valident le test du χ^2.

Peu de résultats de cette section consacrée à l'exposition des principes généraux de statistique seront démontrés dans ce travail. Une démonstration du théorème 34 est tout de même donnée dans la mesure où elle éclaire le résultat et permet ainsi de mieux comprendre l'utilisation du test.

Théorème 34 (test du χ^2)**.** *Avec les notations précédentes, la statistique de Pearson* $D_n^2\left(\hat{\pi}_n, \pi^{\text{ref}}\right)$ *admet le comportement asymptotique suivant :*

- *Sous l'hypothèse* $H_0 : \pi = \pi^{\text{ref}}$, $D_n^2\left(\hat{\pi}_n, \pi^{\text{ref}}\right)$ *converge en loi vers* $\chi^2(d-1)$*, la loi du* χ^2 *à* $d-1$ *degrés de liberté.*
- *Sous l'hypothèse* $H_1 : \pi \neq \pi^{\text{ref}}$, $D_n^2\left(\hat{\pi}_n, \pi^{\text{ref}}\right) \to +\infty$ *presque sûrement.*

Preuve. On commence par l'hypothèse H_0.
Pour tout $t \in [\![1, n]\!]$ on considère la variable aléatoire Z_t définie par :

$$Z_t \overset{\text{def}}{=} \left(\frac{1}{\sqrt{\pi_1^{\text{ref}}}} \left(\mathbb{1}_{\{X_t = x_1\}} - \pi_1^{\text{ref}} \right), \ldots, \frac{1}{\sqrt{\pi_d^{\text{ref}}}} \left(\mathbb{1}_{\{X_t = x_d\}} - \pi_d^{\text{ref}} \right) \right).$$

Par construction, les variables aléatoires Z_1, …, Z_n, sont indépendantes, identiquement distribuées. Leur espérance est le vecteur nul (par construction). Leur loi commune admet un moment d'ordre 2 et la matrice $\Gamma = (\gamma_{i,j}) \in \mathcal{M}_d(\mathbb{R})$ de variance-covariance vérifie :

$$\gamma_{i,j} = \begin{cases} \mathrm{V}\left(\frac{1}{\sqrt{\pi_i^{\text{ref}}}} \left(\mathbb{1}_{\{X_t = x_i\}} - \pi_i^{\text{ref}} \right) \right) = \frac{\pi_i^{\text{ref}} \left(1 - \pi_i^{\text{ref}}\right)}{\pi_i^{\text{ref}}} = 1 - \pi_i^{\text{ref}} & \text{si } i = j \\ \mathrm{Cov}\left(\frac{1}{\sqrt{\pi_i^{\text{ref}}}} \left(\mathbb{1}_{\{X_t = x_i\}} - \pi_i^{\text{ref}} \right), \frac{1}{\sqrt{\pi_j^{\text{ref}}}} \left(\mathbb{1}_{\{X_t = x_j\}} - \pi_j^{\text{ref}} \right) \right) = -\sqrt{\pi_i^{\text{ref}}} \sqrt{\pi_j^{\text{ref}}} & \text{si } i \neq j \end{cases}.$$

En notant $\sqrt{\boldsymbol{\pi}}$, le vecteur défini par $\sqrt{\boldsymbol{\pi}} \overset{\text{def}}{=} \left(\sqrt{\pi_1}, \ldots, \sqrt{\pi_d}\right)$ le résultat du théorème central limite (dans le cas multi-dimensionnel) assure alors la convergence en loi lorsque $n \to \infty$:

$$\frac{1}{\sqrt{n}} \sum_{t=1}^{n} Z_t \rightsquigarrow V,$$

où $V \sim \mathcal{N}\left(0, I_d - {}^t\sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}\right)$. En passant par le carré de la norme euclidienne, on obtient :

$$D_n^2\left(\hat{\pi}_n, \pi^{\text{ref}}\right) \rightsquigarrow \|V\|_2^2.$$

La matrice ${}^t\sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}$ est la matrice associée, dans la base canonique de $\mathbb{R}^d$, à la projection orthogonale sur la droite vectorielle engendrée par le vecteur ${}^t\sqrt{\boldsymbol{\pi}}$.
La matrice $I_d - {}^t\sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}$ est la matrice de la projection orthogonal associée, c'est-à-dire la projection orthogonale sur l'hyperplan $H = \mathrm{Vect}\left({}^t\sqrt{\boldsymbol{\pi}}\right)^{\perp}$ de dimension $d - 1$. Ce qui garantit que V a la même loi que la projection d'un vecteur standard normal sur H. Le résultat du théorème de Cochran assure alors que :

$$\|V\|_2^2 \sim \chi^2(d-1).$$

Ce qui achève la démonstration du premier point du théorème.

Pour démontrer le second point, on se place dans l'hypothèse H_1, on peut donc considérer $i \in [\![1, d]\!]$ tel que $\pi_i \neq \pi_i^{\text{ref}}$. Il suffit alors de remarquer que :

$$D_n^2\left(\hat{\pi}_n, \pi^{\text{ref}}\right) \geqslant n \frac{\left(\hat{\pi}_{i,n} - \pi^{\text{ref}}\right)^2}{\pi_i^{\text{ref}}}.$$

Or, la loi forte des grands nombres assure que :

$$n \frac{\left(\hat{\pi}_{i,n} - \pi^{\text{ref}}\right)^2}{\pi_i^{\text{ref}}} \sim n \underbrace{\frac{\left(\pi_{i,n} - \pi^{\text{ref}}\right)^2}{\pi_i^{\text{ref}}}}_{>0}$$

On a donc finalement :

$$n \frac{\left(\pi_{i,n} - \pi^{\text{ref}}\right)^2}{\pi_i^{\text{ref}}} \xrightarrow[\text{p.s.}]{} +\infty.$$

□

3.2.3 Le test d'autocorrélation

Le test d'autocorrélation consiste à enregistrer les déplacements dans le graphe associé à une chaîne de Markov en leur associant, si les états ne sont pas à valeurs dans $\mathbb{R}$, une valeur numérique.
On note x_i la valeur de l'état atteint au i-ème déplacement (ou avant celui-ci si on tient à prendre en compte dans la formule suivante l'état initial). On calcule alors le coefficient ρ_k d'autocorrélation d'ordre k, défini par :

$$\rho_k \overset{\text{def}}{=} \frac{\sum_{i=1}^{n-k} (x_i - \bar{x})\,(x_{i+k} - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}.$$

Le coefficient ρ_k s'apparente à un calcul de covariance, il est à valeurs dans l'intervalle $[-1, 1]$, on a $\rho_0 = 1$ et on s'attend à ce que ρ_k décroisse avec k croissant. Si les valeurs de ρ_k restent élevées lorsque k augmente, cela indique un temps de mélange long.
Le résultat obtenu dépend du rapport entre le décalage k et la longueur de l'observation des déplacements dans la chaîne n. Si l'utilisation de cet outil s'appuie sur l'art et l'expérience du statisticien on peut se convaincre de son efficacité en observant les résultats obtenus pour la chaîne paresseuse sur un 20-cycle pour différentes valeurs de n représentés sur la figure 3.10.

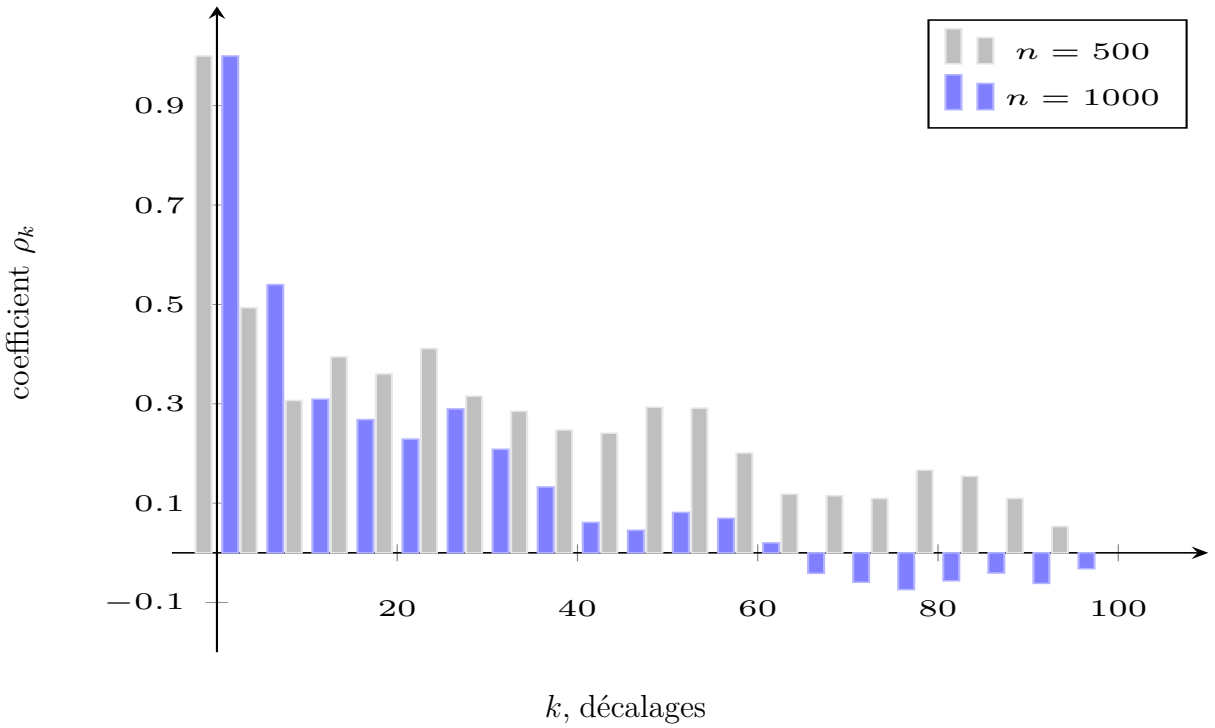


FIGURE 3.10 – Résultats du test d'autocorrélation pour la chaîne paresseuse sur un 20-cycle

3.2.4 Le test de Gelman-Rubin

Le principe du test de Gelman-Rubin est le suivant :

1. On lance m chaînes (au moins deux) de même longueur $2n$ partant de m points de départ distincts.
2. On rejette les n premières étapes de chaque chaîne.

3. On calcule W la moyenne des variances empiriques de chaque chaîne pour le paramètre θ considéré (*within-chain variance*) :

$$W \stackrel{\text{def}}{=} \frac{1}{m} \sum_{j=1}^{m} s_j^2$$

où :

$$s_j^2 \stackrel{\text{def}}{=} \frac{1}{n-1} \sum_{i=1}^{n} \left(\theta_{ij} - \overline{\theta}_j\right)^2$$

θ_{ij} est la i-ième valeur du paramètre pour la chaîne j, $\overline{\theta}_j$ est la moyenne du paramètre pour la chaîne j. Le facteur $1/(n-1)$ plutôt que $1/n$ permet d'avoir un estimateur non biaisé

4. On calcule la variance B "inter chaînes" (*between-chain variance*) :

$$B \stackrel{\text{def}}{=} \frac{n}{m-1} \sum_{j=1}^{m} \left(\overline{\theta}_j - \overline{\overline{\theta}}\right),$$

où $\overline{\overline{\theta}}$ est la moyenne de tous les paramètres obtenus sur toutes les chaînes.
B mesure donc la variance des moyennes obtenues pour chaque chaîne par rapport à cette moyenne globale.

5. On calcule la variance estimée $\widehat{\text{Var}}$ du paramètre à partir d'une moyenne pondérée de W et B :

$$\widehat{\text{Var}} \stackrel{\text{def}}{=} \left(1 - \frac{1}{n}\right) W + \frac{1}{n} B.$$

Le "poids" de B dans $\widehat{\text{Var}}$ est important lorsque les chaînes sont encore éloignées de la distribution stationnaire.

6. On calcule alors le coefficient $\widehat{R}$ (*potential scale reduction factor*), défini par :

$$\widehat{R} \stackrel{\text{def}}{=} \sqrt{\frac{\widehat{\text{Var}}}{W}} = \sqrt{\left(1 - \frac{1}{n}\right) + \frac{1}{n}\frac{B}{W}}$$

Ce coefficient permet de détecter un "poids" de B trop important dans l'évaluation de $\widehat{\text{Var}}$.
En pratique $\widehat{R}$ supérieur à 1.1 ou 1.2 dénote un mauvais mélange.

Pour illustrer le propos, la figure 3.11 donne les valeurs de $\widehat{R}$ pour la marche aléatoire paresseuse sur un 20-cycle pour différentes valeurs de m :

- $m = 2$, en prenant une chaîne commençant en 1 et l'autre en 11.
- $m = 3$, en prenant trois chaînes commençant respectivement en 1, 7 et 14.
- $m = 20$, en faisant démarrer une chaîne en chacun des sommets du cycle. Cet exemple est bien entendu très artificiel puisqu'en pratique m est très inférieur au cardinal de l'ensemble des états.

Il faut bien noter que comme souvent en statistique ce test est assez valable pour invalider un nombre de déplacements trop petits dans le graphe associé à la chaîne. Une valeur de $\widehat{R}$ proche de 1 permet d'accorder une confiance au nombre de déplacements choisis, elle ne "démontre" pas que le temps de mélange est atteint.

FIGURE 3.11 – Évolution du *potential scale reduction factor* $\widehat{R}$ pour différentes valeurs de m, les premières valeurs sont calculées pour $n = 20$, puis de 20 en 20 jusqu'à $n = 1000$

Deuxième partie

Contributions

Chapitre 4

Génération aléatoire d'automates à pile déterministes, temps réel

Sommaire

4.1 Introduction

Les automates à pile déterministes temps réel, abrégés en APDTR, ont été introduits à la section 1.3. On rappelle ici qu'un automate à pile déterministe à temps réel est un 7-uplet $(Q, \Sigma, \Gamma, Z_{init}, \delta, q_{init}, F)$ où Q désigne l'ensemble fini des états, Σ est l'alphabet de mots, Γ l'alphabet de pile, ces deux alphabets sont disjoints, q_{init} est un élément de Q qui désigne l'état initial, F est le sous-ensemble, éventuellement vide de Q des états finals, Z_{init} est le symbol initial de la pile, δ est une fonction de $Q \times (\Sigma \times \Gamma)$ dans $Q \times \Gamma^*$. On rappelle également que l'on dit que le quadruplet de la forme $(q, (a, X), \omega, p) \in Q \times (\Sigma \times \Gamma) \times \Gamma^* \times Q$ est une *transition* dans le cas où $\delta((q, (a, X))) = (p, \omega)$. Si c'est le cas, on dit que ω est le mot de sortie de la transition, la *taille de sortie* de la transition est la longueur du mot ω. La *taille de sortie* de l'APDTR est la somme des tailles de sortie de toutes les transitions. La section 1.3 formalise également les notions d'*automate sous-jacent* à un APDTR, d'*accessibilité*, de *pop-transition*, de *configuration*, de configurations *consécutives*, d'état *atteignable*.

Dans la section 4.2, on donne un résultat asymptotique sur le nombre d'automates à pile déterministes et complets. La section 4.3 explique succinctement le principe du générateur. On utilise le générateur afin de réaliser des expériences sur l'influence de la condition d'acceptation dans la section 4.4.

4.2 Énumération des APDTR

4.2.1 Principe du calcul

Le problème consiste ici à compter, à isomorphisme près, le nombre des APDTR à n états, construits sur les alphabets Σ et Γ, comptant s transitions, tels que la taille de sortie soit égale à m (on parlera alors de APDTR de taille m). On notera dans la suite $\mathfrak{T}_{s,n,m}$ la classe combinatoire correspondante.

Le principe s'appuie sur l'énumération préalable des automates sous-jacents associés. Ils s'agit d'automates déterministes à n états et s transitions, construits sur l'alphabet $\Sigma \times \Gamma$. On note $\mathfrak{A}_{s,n}$ la classe combinatoire de ces automates, eux aussi définis à isomorphisme près. Il s'agit ensuite de déterminer le nombre de façons d'équiper chacun de ces automates en opérations de pile afin de construire un APDTR dont la taille des opérations de pile soit égale à m.

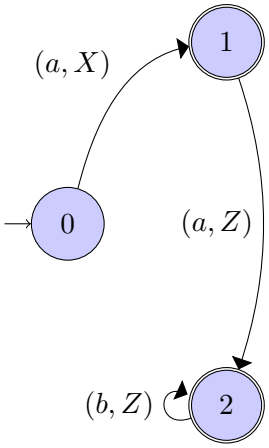
Formellement, en notant ψ l'application de $\mathfrak{T}_{s,n,m}$ dans $\mathfrak{A}_{s,n}$ qui associe à un APDTR son automate sous-jacent, pour tout $\mathcal{A} \in \mathfrak{A}_{s,n}$, le cardinal de l'image réciproque $\psi^{-1}(\{\mathcal{A}\})$ est le nombre d'étiquetages possibles des s transitions de $\mathcal{A}$ de telle sorte que la taille de sortie des éléments de $\psi^{-1}(\{\mathcal{A}\})$ soit égale à m. Ce cardinal est indépendant de la structure de $\mathcal{A}$, il ne dépend que des deux nombres entiers s et m ; on le note $c_{s,m}$. La proposition suivante découle directement des remarques précédentes :

Proposition 35 (énumération des APDTR)**.** *Avec les notations précédentes, on a :*

$$|\mathfrak{T}_{s,n,m}| = c_{s,m} \cdot |\mathfrak{A}_{s,n}| \,. \tag{4.1}$$

On exploitera l'équation 4.1 de la proposition 35 grâce au résultat du théorème 6 de la section 4.2.3. Ce résultat, présenté dans [Kor78], permet de donner une évaluation asymptotique de $|\mathfrak{A}_{s,n}|$. On pourra sur le sujet se référer également à [BN07].
Une première étape consiste à déterminer $c_{s,m}$. La section suivante est consacrée à ce calcul.

La figure 4.1 illustre cette idée en représentant un automate sous-jacent à 3 états et 3 transitions sur l'alphabet $\Sigma \times \Gamma \stackrel{\text{def}}{=} \{a, b\} \times \{Z, X\}$, ainsi que les 24 étiquetages possibles pour une taille des opérations de pile égale à 2.



(a,X)	(a,Z)	(b,Z)
ε	ε	XX
ε	ε	XZ
ε	ε	ZX
ε	ε	ZZ
ε	XX	ε
ε	XZ	ε
ε	ZX	ε
ε	ZZ	ε
XX	ε	ε
XZ	ε	ε
ZX	ε	ε
ZZ	ε	ε

(a,X)	(a,Z)	(b,Z)
ε	X	X
ε	X	Z
ε	Z	X
ε	Z	Z
X	ε	X
X	ε	Z
Z	ε	X
Z	ε	Z
X	X	ε
X	Z	ε
Z	X	ε
Z	Z	ε

FIGURE 4.1 – Un automate sous-jacent, étiquetages des opérations de pile pour une taille égale à 3

4.2.2 Calcul de $c_{s,m}$

Le calcul de $c_{s,m}$ constitue un problème classique de combinatoire. La fonction génératrice associée à la classe combinatoire des opérations de pile est facile à déterminer. Si la fonction génératrice est connue, les coefficients du développement peuvent être difficiles à exprimer, ce n'est pas le cas ici comme le montrent les considérations qui suivent.

Définir une transition d'un automate $\mathcal{A}$ sous-jacent consiste à associer à un état q et une lettre (a, X) de l'alphabet $\Sigma \times \Gamma$ un état. L'ensemble des APDTR qui admettent le même automate sous-jacent $\mathcal{A}$ est l'ensemble $\psi^{-1}(\{\mathcal{A}\})$. Pour définir les transitions d'un APDTR élément de $\psi^{-1}(\{\mathcal{A}\})$, il faut associer à q, a et X, en plus d'un état, une opération de pile, c'est-à-dire un mot, éventuellement vide, de l'alphabet Γ. Il s'agit donc de déterminer s mots sur l'alphabet Γ dont la somme des longueurs est égale à m et d'attribuer chacun de ces s mots à une des s transitions. En supposant les ensembles Q, Σ et Γ totalement ordonnés, on peut ordonner les s transitions de $\mathcal{A}$ (dans l'ordre lexicographique des triplets (q, a, X) par exemple), $c_{s,m}$ est donc le nombre de s-uplets $(\omega_1, \ldots, \omega_s)$ de Γ^* tels que $\sum |\omega_k| = m$.

En notant $\mathcal{M}$ la classe combinatoire des mots construits sur l'alphabet Γ et β le cardinal de Γ, la fonction génératrice ordinaire M associée à $\mathcal{M}$ vérifie :

$$M(z) = \sum_{n \geqslant 0} (\beta z)^n = \frac{1}{1 - \beta z}, \tag{4.2}$$

en effet, pour tout nombre entier n, il existe β^n mots de longueur n,

La fonction génératrice ordinaire associée à une liste de s mots est donc $M(z)^s$, on a donc :

$$c_{s,m} = [z^m]\, M(z)^s. \tag{4.3}$$

En posant $F_s(z) \stackrel{\text{def}}{=} M(z)^s$, la fonction génératrice F_s vérifie :

$$F_s' = s \cdot \beta \cdot F_{s+1}. \tag{4.4}$$

Par récurrence immédiate à partir de l'équation 4.4, on établit que pour tout nombre entier naturel m :

$$F_s^{(m)} = \beta^m s(s+1)(s+2)\ldots(s+m-1)F_{s+m}. \tag{4.5}$$

La formule de Taylor permet alors d'obtenir :

$$c_{s,m} = [z^m]\, F_s(z) = \beta^m \frac{s(s+1)(s+2)\ldots(s+m-1)}{m!} = \beta^m \binom{s+m-1}{m}. \tag{4.6}$$

4.2.3 Asymptotique de $|\mathfrak{A}_{\rho n,n}|$

Il est naturel de s'intéresser aux automates pour lesquels le nombre de transitions est proportionnel au nombre d'états. C'est le cas par exemple pour les automates complets.

L'automate à pile, comme son automate sous-jacent, est complet si et seulement si le nombre de s de transitions vérifie $s = n \cdot |\Sigma| \cdot |\Gamma|$.

En notant α le cardinal de Σ, les automates complets sont donc ceux pour lesquels $s = \alpha \cdot \beta \cdot n$. Dans la suite on posera $\rho \stackrel{\text{def}}{=} \alpha\beta$.

Dans ce cadre, le théorème 6 de la section 1.1.5 donne une estimation de l'énumération des éléments de $\mathfrak{T}_{s,n,m}$. On rappelle ici le résultat :

Théorème 36 ([Kor78]). *Il existe un nombre réel positif ne dépendant que de ρ, γ_ρ, tel que :*

$$|\mathfrak{A}_{\rho n,n}| \sim n \cdot \gamma_\rho \cdot \left\{ \begin{matrix} \rho n \\ n \end{matrix} \right\}.$$

où $\left\{ \begin{matrix} \rho n \\ n \end{matrix} \right\}$ désigne le nombre de Stirling de seconde espèce associé à ρ et n.

La figure 4.2 donne une idée des valeurs de $n \cdot \left\{ \begin{matrix} \rho n \\ n \end{matrix} \right\}$ en fonction de n pour différentes valeurs de ρ. La croissance est exponentielle, l'échelle est donc logarithmique.

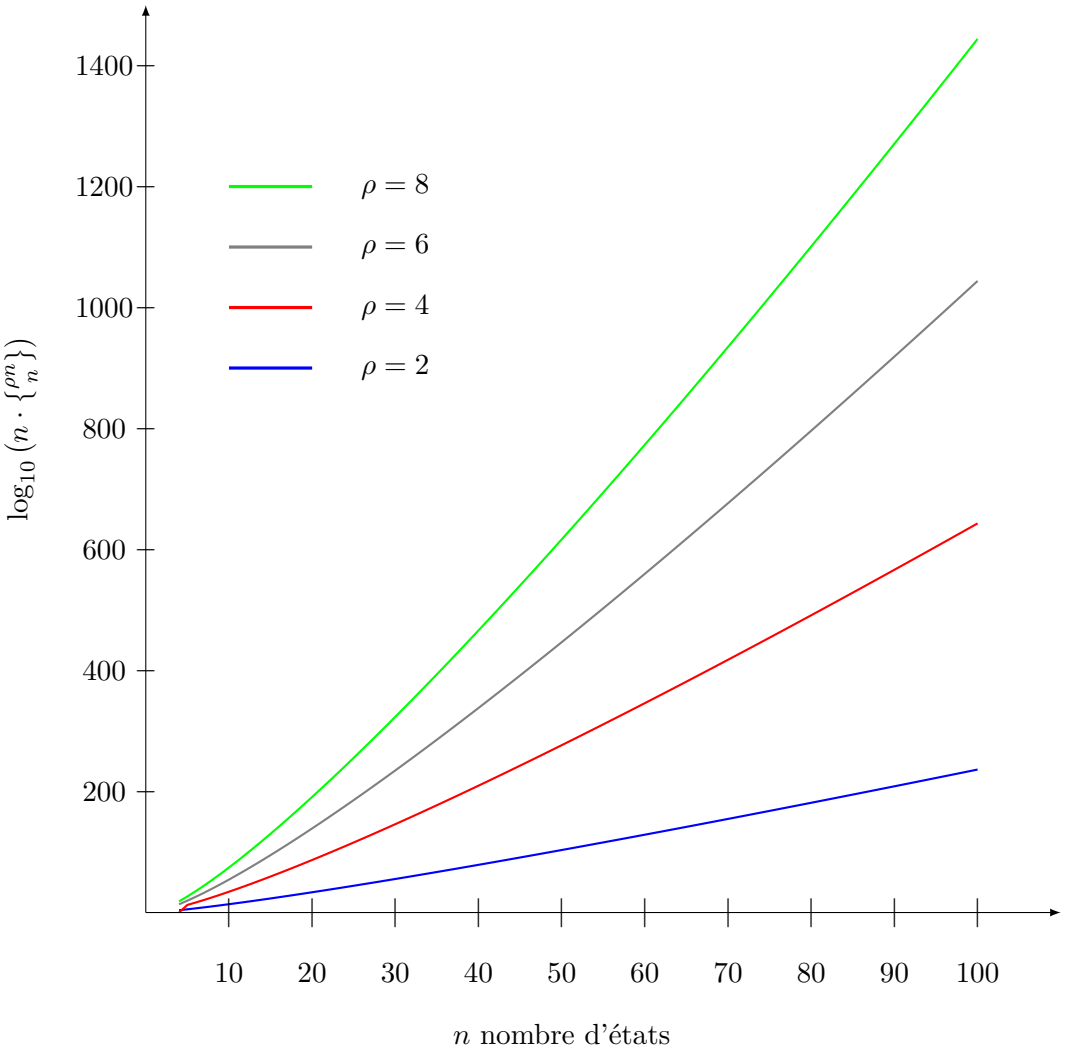


FIGURE 4.2 – Valeurs de $\log_{10}\left(n \cdot \left\{ \begin{matrix} \rho n \\ n \end{matrix} \right\}\right)$ pour différentes valeurs de ρ

4.2.4 Asymptotique de $c_{m,s}$

Ici encore la démarche consiste à considérer les paramètres du problème proportionnels. On suppose donc ici que la taille de sortie m vérifie $m = \lambda \cdot s$, où λ est un nombre rationnel strictement positif tel que $\lambda \cdot s$ soit un nombre entier. Deux démarches conduisant au même résultat sont alors envisageables. La première utilise le théorème de point col 16 exposé à la section 2.5 de la page 35, la seconde s'appuie uniquement sur la formule de Stirling. La première méthode a un caractère plus général, dans la mesure où la formule de Stirling ne peut évidemment s'appliquer qu'à des formules adaptées. La convergence des deux méthodes vers le même résultat illustre cependant de façon typique le contexte d'utilisation de ces techniques de col.

Méthode avec technique de col

L'application M définie par l'égalité 4.2 vérifie les propriétés suivantes :

- M est analytique en 0.
- Le développement en série de M ne compte que des coefficients positifs,
- M n'est pas nulle en 0.
- M est apériodique si l'alphabet Γ compte au moins deux lettres.
- Le rayon de convergence R de M est égal à $\frac{1}{\beta}$ et :

$$x\frac{M'(x)}{M(x)} = \frac{x\beta}{1-\beta x}$$

donc :

$$\lim_{x\to R^-} x\frac{M'(x)}{M(x)} = +\infty$$

On peut donc appliquer le théorème 16 de la section 2.5 et affirmer que pour tout nombre réel strictement positif λ on a :

$$\left[z^{\lambda\cdot s}\right] F_s(z) \underset{s\to\infty}{\sim} \frac{M(\zeta)^s}{\zeta^{\lambda\cdot s}\sqrt{2\pi\cdot s\cdot\xi}}, \tag{4.7}$$

où $\zeta = \frac{\lambda}{\beta(1+\lambda)}$ est l'unique solution de l'équation $x\frac{M'(x)}{M(x)} = \lambda$. Notons que $\zeta \in]0, 1/\beta[$.
Par ailleurs, ξ est la valeur prise par l'expression :

$$\frac{d^2}{dz^2}\left(\log\left(M(z)\right) - \lambda\log(z)\right), \tag{4.8}$$

pour $z = \zeta$. On a donc :

$$\xi = \frac{(\lambda+1)^3\beta^2}{\lambda}. \tag{4.9}$$

L'équation 4.7 peut donc se ré-écrire en fonction uniquement des valeurs de s, λ et β :

$$\left[z^{\lambda\cdot s}\right] F_s(z) \underset{s\to\infty}{\sim} \beta^{\lambda\cdot s}\frac{(\lambda+1)^{(\lambda+1)s-1/2}}{\lambda^{\lambda\cdot s+1/2}\sqrt{2\pi\cdot s}}. \tag{4.10}$$

Par la formule de Stirling

Dans le développement qui suit on utilise le résultat du lemme 37, ce résultat n'est qu'une ré-écriture de l'équivalent classique : $\left(1-\frac{1}{n}\right)^n \underset{n\to\infty}{\sim} \frac{1}{e}$.

Lemme 37. *On a :*

$$(n-1)^{n-1} \underset{n\to\infty}{\sim} \frac{n^n}{e\cdot n}$$

Le point de départ est une réécriture de la formule 4.6 tenant compte de l'égalité $m = \lambda\cdot s$:

$$\left[z^{\lambda\cdot s}\right] F_s(z) = \beta^{\lambda\cdot s}\binom{(\lambda+1)s-1}{\lambda\cdot s} \tag{4.11}$$

Or :

$$\binom{(\lambda+1)s-1}{\lambda\cdot s} = \frac{((\lambda+1)s-1)!}{(\lambda\cdot s)!(s-1)!} \tag{4.12}$$

et :

$$\frac{((\lambda+1)s-1)!}{(\lambda\cdot s)!(s-1)!} \underset{s\to\infty}{\sim} \frac{((\lambda+1)s-1)^{((\lambda+1)s-1)}}{(\lambda\cdot s)^{(\lambda\cdot s)}\,(s-1)^{(s-1)}}\frac{\sqrt{(\lambda+1)s-1}}{\sqrt{2\pi\lambda s(s-1)}}. \tag{4.13}$$

Or :

$$\frac{\sqrt{(\lambda+1)s-1}}{\sqrt{2\pi\lambda s(s-1)}} \underset{s\to\infty}{\sim} \frac{1}{\sqrt{2\pi\cdot s}}\left(\frac{\lambda+1}{\lambda}\right)^{1/2}. \tag{4.14}$$

Le résultat du lemme 37 permet d'établir que :

$$\frac{((\lambda+1)s-1)^{((\lambda+1)s-1)}}{(\lambda\cdot s)^{(\lambda\cdot s)}\,(s-1)^{(s-1)}} \underset{s\to\infty}{\sim} \frac{(\lambda+1)^{(\lambda+1)s}}{\lambda^{\lambda\cdot s}(\lambda+1)}. \tag{4.15}$$

Les équations 4.11, 4.14 et 4.15, permettent alors d'établir que

$$\left[z^{\lambda\cdot s}\right] F_s(z) \underset{s\to\infty}{\sim} \beta^{\lambda\cdot s}\frac{(\lambda+1)^{(\lambda+1)s}}{\lambda^{\lambda\cdot s}(\lambda+1)}\frac{1}{\sqrt{2\pi\cdot s}}\left(\frac{\lambda+1}{\lambda}\right)^{1/2}, \tag{4.16}$$

qui n'est qu'une reformulation du résultat 4.10.

4.2.5 Asymptotique de $|\mathfrak{T}_{s,n,m}|$

Le résultat de la proposition 38 se déduit directement des résultats de la proposition 35, du théorème 6 et des équations 4.10 et 4.16. Dans le cas d'automates complets, la proposition rappelle les relations de proportionnalité entre les différents paramètres.

Proposition 38 (Énumération des APDTR complets)**.** *On considère $f_{\lambda,n,\alpha,\beta}$ le nombre d'automates APDTR complets à n états, munis d'un alphabet Σ de cardinal α, d'un alphabet de pile Γ de cardinal β, tels que la taille de sortie m vérifie $m = \lambda\cdot\rho\cdot n$, où $\rho \stackrel{def}{=} \alpha\cdot\beta$ et λ est un nombre rationnel strictement positif tel que $\lambda\cdot\rho\cdot n \in \mathbb{N}$. Alors :*

$$f_{\lambda,n,\alpha,\beta} \underset{n\to\infty}{\sim} n\cdot\gamma_\rho\cdot\left\{\begin{matrix}\rho n\\ n\end{matrix}\right\}\cdot\beta^{\lambda\cdot\rho\cdot n}\frac{(\lambda+1)^{(\lambda+1)\cdot\rho\cdot n-1/2}}{\lambda^{\lambda\cdot\rho\cdot n+1/2}\sqrt{2\pi\cdot\rho\cdot n}}.$$

La figure 4.3 montre que l'évaluation de $\binom{\rho n+\lambda\rho n-1}{\lambda\rho n}$ par des fonctions exponentielles est pertinente même pour de petites valeurs de n.

4.2.6 Nombre moyen de *pop-transitions*

Nous nous intéresserons dans la section 4.4 à différents modes d'acceptation d'un mot par les automates à pile. La reconnaissance d'un langage par pile vide est un des modes possibles d'acceptation. La pile se vide grâce aux *pop-transitions*, l'évaluation du nombre moyen de *pop-transitions* dans les APDTR constitue donc un point d'intérêt. Le calcul du nombre moyen de *pop-transitions* passe par ailleurs par des techniques dérivées des techniques d'énumération utilisées précédemment.

En reprenant les notations précédentes, il s'agit donc d'obtenir le nombre moyen de mots vides dans une liste de s mots construits sur un alphabet Γ de β lettres lorsque la somme m des longueurs des mots de la liste est fixée.

Dans la série génératrice ordinaire $M(z) = 1+\sum_{n>0}(\beta z)^n$ associée aux mots construits sur l'alphabet Γ, où β est le cardinal de l'alphabet Γ, on marque par u le mot vide, on obtient la fonction génératrice ordinaire bivariée :

$$u + \sum_{n>0}(\beta z)^n.$$

En notant $G_s(z,u)$ la série génératrice ordinaire associée aux listes de s mots construits sur l'alphabet Γ, on obtient :

$$G_s(z,u) = \left(u+\sum_{n>0}(\beta z)^n\right)^s = \left(\frac{1}{1-\beta z}-1+u\right)^s.$$

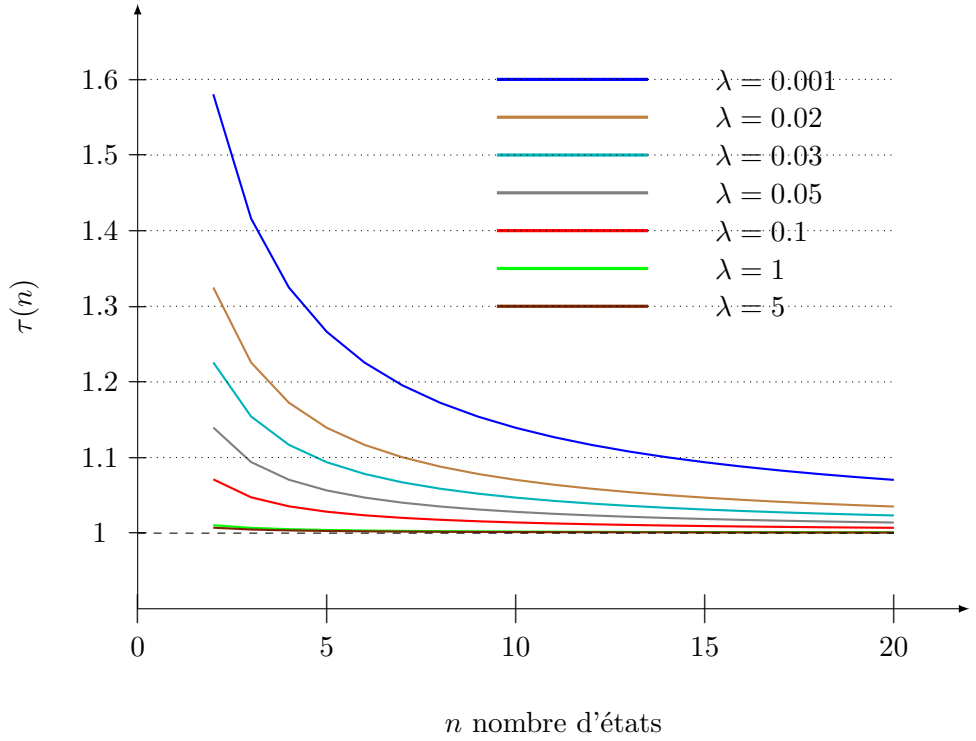


FIGURE 4.3 – Évaluation du rapport $\tau(n) = \dfrac{(\lambda+1)^{(\lambda+1)\rho n-1/2}}{\lambda^{\lambda\rho n+1/2}\sqrt{2\pi\rho n}}$ sur $\binom{\lambda n\rho+n\rho-1}{\lambda\rho n}$, pour différentes valeurs de λ en fonction de n, le nombre d'états de l'automate à pile considéré. $\rho = 6$ et $\beta = 3$

Un résultat classique de combinatoire [FS08, Proposition III.2] permet alors de déterminer le nombre moyen μ_ε de mots vides dans une liste de s mots construits sur l'alphabet Γ :

$$\mu_\varepsilon = \frac{[z^m]\,\frac{\partial}{\partial u}G_s(z,u)_{|u=1}}{[z^m]\,G_s(z,1)}.$$

On a :

$$\frac{\partial}{\partial u}G_s(z,u) = s\cdot G_{s-1}(z,u).$$

Par ailleurs :

$$\frac{\partial}{\partial z}G_{s-1}(z,u) = (s-1)\cdot\beta\cdot G_{s-1}(z,u).$$

Ce qui permet d'obtenir par récurrence immédiate :

$$\begin{aligned}\frac{\partial^m}{\partial z^m}G_s(z,u) &= \beta^m s(s+1)\dots(s+m-1)G_{s+m}(z,u)\\ \frac{\partial^m}{\partial z^m}G_{s-1}(z,u) &= \beta^m (s-1)s(s+1)\dots(s+m-2)G_{s+m-1}(z,u).\end{aligned}$$

Alors :

$$\begin{aligned}[z^m]\,G_s(z,1) &= [z^m]\,\beta^m\frac{s(s+1)\dots(s+m-1)}{m!}\\ [z^m]\,\frac{\partial}{\partial u}G_s(z,u)_{|u=1} &= [z^m]\,s\cdot G_{s-1}(z,u)_{|u=1} = \beta^m\frac{(s-1)s^2(s+1)\dots(s+m-2)}{m!}.\end{aligned}$$

On a donc :

$$\mu_\varepsilon = \frac{s(s+1)\dots(s+m-1)}{(s-1)s^2(s+1)\dots(s+m-2)} = \frac{s(s-1)}{s+m-1}.$$

Ce qui permet d'établir le résultat de la proposition 39 :

Proposition 39 (Nombre moyen de *pop-transitions*)**.** *Le nombre moyen de* pop-transitions *d'un APDTR à n états, s transitions pour une taille de sortie m fixée est :*

$$\frac{s(s-1)}{s+m-1}.$$

Ce résultat appelle plusieurs remarques :

- Lorsque la taille m de sortie et le nombre s de transitions des APDTR considérés sont proportionnels au nombre n d'états, alors le nombre moyen μ_ε de *pop-transitions* vérifie $\mu_\varepsilon \underset{n\to\infty}{\sim} \Theta(n)$.
- Le résultat de la proposition 39 confirme l'intuition que pour un nombre s de transitions fixé, si la taille de sortie m est proportionnelle au nombre d'états, alors le nombre moyen μ_ε de *pop-transitions* vérifie $\mu_\varepsilon \underset{n\to\infty}{=} o(1)$.

4.3 Génération aléatoire des APDTR

Le principe général de génération aléatoire uniforme des APDTR est basé sur le résultat de la proposition 35. On procède en deux temps :

1. On tire uniformément un élément $\mathcal{A}$ de $\mathfrak{A}_{s,n}$.
2. On tire uniformément un s-uplet de mots de Γ^* tels que la somme des longueurs de ces mots soit égale à m.

La première étape est réalisée en utilisant [HNS10].

Pour les APDTR complets, lorsque le nombre de transitions s est proportionnel au nombre n d'états, des algorithmes plus efficaces existent et sont exposés dans [BDN07] et [CN12]. Dans le cas général, la complexité est $\mathcal{O}(n^3)$ et dans le cas des APDTR complets, la complexité passe à $\mathcal{O}\left(n^{\frac{3}{2}}\right)$.

La seconde étape, quant à elle, est réalisée par une approche récursive classique comme décrit dans [FS08] ou par un générateur de Boltzmann. Le générateur de Boltzmann est linéaire en la taille des opérations de pile souhaitée.

4.4 Influence de la condition d'acceptation

À propos des états d'un APDTR, il est nécessaire de bien rappeler ici les notions d'accessibilité et d'atteignabilité exposées dans la section 1.3.2. Un état p est accessible s'il l'est au sens de l'automate de mots sous-jacent, il est atteignable s'il existe une suite de configurations consécutives $(p_1, \omega_1), \ldots, (p_n, \omega_n)$, partant de la configuration initiale, telle que $p_n = p$. Tout état atteignable est accessible, la réciproque n'est pas vraie dans le cas général.

On rappelle également brièvement les différents mode d'acceptabilité d'un mot :

- Par pile vide.
- Par état final.
- Par pile vide et état final.

La génération aléatoire de APDTR peut donc conduire à des automates associés à des langages vides ou comportant des états inatteignables, donc inutiles. On peut donc raisonnablement imposer à un APDTR obtenu par le générateur, l'une des exigences suivantes :

- reconnaissance d'un langage non-vide,
- tous les états sont atteignables,
- tous les états finals sont atteignables par pile vide (dans le cas de l'acceptation par pile vide et état final).

4.4.1 À propos des APDTR reconnaissant un langage non-vide

En reprenant les notations précédentes portant sur les cardinaux des alphabets Σ et Γ, la proposition 40 donne un minorant de la probabilité d'obtenir un APDTR acceptant un langage non-vide.

Proposition 40 (langage vide, minoration)**.** *Quelle que soit la condition d'acceptation, la probabilité qu'un APDTR accessible possédant au moins deux états, s transitions, de taille de sortie m telle que $m \geqslant 1$, reconnaisse un langage non-vide est supérieure ou égale à* $\dfrac{s-1}{2\beta(s+m-1)}$

Preuve. Puisque les APDTR considérés sont accessibles et possèdent au moins deux états, il existe au moins une transition τ partant de l'état initial q_{init}. La preuve de la proposition s'établit en calculant la probabilité que la transition τ soit de la forme $(q_{\text{init}}, (a, Z_{\text{init}})) \to (\varepsilon, p)$ où p est un état final. La lettre a de l'alphabet Σ n'influe pas sur la probabilité cherchée. La probabilité que le symbole de pile de la transition τ soit Z_{init} est $\frac{1}{\beta}$, les β lettres de l'alphabet Γ étant équiprobables. La probabilité que l'état p soit final est égale à $\frac{1}{2}$ (pour chaque état, une fois l'automate engendré, on tire avec une probabilité $\frac{1}{2}$ s'il est final ou non, voir [BDN07, CP05]). Le résultat de la proposition 39 garantit que τ est une *pop-transition* avec la probabilité $\frac{s-1}{s+m-1}$. La transition τ est donc de la forme $(q_{\text{init}}, (a, Z_{\text{init}})) \to (\varepsilon, p)$, avec p final, avec la probabilité $\frac{s-1}{2\beta(s+m-1)}$. Si c'est le cas, l'automate reconnaît alors le mot a. □

Si comme dans les calculs menés dans l'étude asymptotique de l'énumération des APDTR, on considère une taille de sortie des APDTR proportionnelle au nombre s de transitions, le résultat de la proposition 40 assure que la probabilité d'obtenir un APDTR reconnaissant un langage non-vide est supérieure à une fonction positive croissante du nombre de transitions s, qui tend vers $\frac{1}{2\beta(\lambda+1)}$ lorsque s tend vers l'infini. On rappelle que le coefficient λ est défini par $\lambda \stackrel{\text{def}}{=} \frac{m}{s}$ et s'interprète donc comme la taille moyenne des opérations de pile par transition. Il est donc possible d'utiliser un algorithme de rejet efficace afin de ne garder que des APDTR reconnaissant un langage non-vide.

4.4.2 À propos de la reconnaissance par pile vide

Le résultat de la proposition 40 assure qu'il est possible d'engendrer des APDTR complets qui reconnaissent un langage non-vide dans le cas où on impose également au nombre s de transitions d'être proportionnel à la taille de sortie des automates.

Cela ne règle pas cependant la question des états qui ne sont pas atteignables. Si un mot est reconnu par pile vide et état final, un état final qui n'est pas atteignable avec la pile vide est un état inutile dans la mesure où il ne peut pas être utilisé pour reconnaître un mot. Néanmoins, cet état peut intervenir dans une suite de transitions conduisant à la reconnaissance d'un mot, en terminant par pile vide sur un autre état final. Les outils pour déterminer en temps polynomial si un état est atteignable avec une pile vide existent [FWW97]. Le tableau 4.1 rassemble les résultats des expériences qui ont été menées sur le nombre moyen d'états (finals ou non) atteignables par pile vide pour des APDTR complets, accessibles, pour des alphabets Σ et Γ de même cardinal : $\alpha = \beta = 2$.

Pour chaque entrée de ce tableau 100 APDTR ont été engendrés.

Puisque le caractère final ou non d'un état est issu d'un tirage de Bernoulli équilibré, on obtient le nombre moyen d'états finals atteignables par pile vide, en divisant les chiffres du tableau par 2.

Les résultats des expériences montrent que pour $\lambda \geqslant 1$, le nombre moyens d'états finals atteignables par pile vide est relativement faible.

nombre d'états →	5	10	15	20	30	40	60	100
$\lambda = 0.5$	3.56	6.14	8.02	11.1	16.26	15	24.7	49.5
$\lambda = 1$	2.6	4.62	4.7	6.16	7.06	7.82	13.85	17.3
$\lambda = 1.5$	2.36	3.05	3.61	3.62	5.2	5.5	5.68	5.8
$\lambda = 2$	2.0	2.6	2.81	3.4	3.02	3.1	3.21	3.89
$\lambda = 3$	1.65	1.8	1.83	1.8	2.26	2.44	2.34	2.6
$\lambda = 5$	1.3	1.41	1.43	1.4	1.42	2.1	1.5	1.5

TABLE 4.1 – Nombre moyen d'états atteignables avec une pile vide $\alpha = 2$, $\beta = 2$

Ces premiers résultats confirment que la production aléatoire de APDTR complets et accessibles, basée sur la taille de sortie, ne fournira pas suffisamment de *pop-transitions* pour vider la pile. Une idée consiste à imposer un nombre minimal k de *pop-transitions* . Bien sûr, cela change la distribution de génération puisque le nombre de *pop-transitions* est imposé. Cette idée peut être menée à bien de la façon suivante :

1. Tirer aléatoirement et uniformément k transitions de l'automate sous-jacent à qui on imposera d'être des *pop-transitions* du APDTR engendré.
2. Compléter les $s - k$ autres transitions afin d'obtenir une taille de sortie égale à m.

Les mêmes expériences que celles qui ont conduit au tableau 4.1 ont été menées afin de pouvoir comparer les résultats. On a donc encore choisi $\alpha = \beta = 2$ mais en imposant au moins 60% de *pop-transitions* . Les résultats sont donnés dans le tableau 4.2. Ils montrent qu'imposer un nombre minimal de *pop-transitions* améliore beaucoup la proportion d'états atteignables pour les petites valeurs de λ, cette proportion reste insatisfaisante pour $\lambda \geqslant 1$

nombre d'états →	5	10	15	20	30	40	60	100
$\lambda = 0.5$	3.53	6.9	9.43	12.23	17.95	23.72	35.97	62.45
$\lambda = 1$	3.09	5.49	7.64	10.53	14.23	18.92	28.02	50.02
$\lambda = 1.5$	3.23	5.31	7.67	9.88	12.27	15.32	23.53	35.61
$\lambda = 2$	2.78	4.81	6.25	8.22	12.36	14.71	23.43	33.03
$\lambda = 3$	3.08	5.01	6.31	7.87	10.76	14.16	18.57	30.17
$\lambda = 5$	2.72	4.09	5.73	6.54	9.42	12.36	19.65	

TABLE 4.2 – Nombre moyen d'états atteignables par pile vide (proportion de *pop-transitions* $\geqslant 60\%$) ; $\alpha = \beta = 2$.

4.4.3 À propos de la reconnaissance par état final exclusivement

Puisque l'état de la pile n'a plus d'influence sur la reconnaissance des mots, il s'agit ici de compter le nombre moyen d'états atteignables, quel que soit l'état de la pile. Encore une fois, le nombre moyen d'états finals atteignables s'obtient simplement en divisant les résultats par 2. Les outils proviennent des mêmes références [FWW97], ils ont été utilisés pour établir les résultats du tableau 4.3 pour lequel $\alpha = \beta = 2$ et du tableau 4.4 pour lequel $\alpha = 3$ et $\beta = 5$.

Ces derniers résultats valident le cadre proposé pour la génération aléatoire uniforme de APDTR accessibles et complets lorsque que l'acceptation n'est réalisée que par état final. En effet, la plupart des états sont atteignables. Pour les autres conditions d'acceptation, la valeur de λ doit ne pas être trop grande afin d'assurer un nombre suffisant de *pop-transitions* pour vider la pile. Cette observation est à rapprochée du nombre moyen de *pop-transitions* donné par la proposition 39. Dans le cas complet, lorsque le nombre d'états tend vers l'infini, ce nombre moyen est de l'ordre de $\frac{\alpha\beta n}{\lambda+1}$.

nombre d'états →	10	20	30	40	50	60	80	100
$\lambda = 1$	8.29	14.89	21.5	26.93	32.48	35.55	44.4	52.86
$\lambda = 2$	8.73	16.35	25.3	33.45	39.56	47.99	62.32	81.02
$\lambda = 3$	8.84	17.67	27.14	36.19	45.7	54.69	73.35	89.26
$\lambda = 5$	9.23	18.3	28.06	37.47	47.61	56.7	76.11	95.15

TABLE 4.3 – Nombre moyen d'états atteignables, $\alpha = \beta = 2$.

nombre d'états →	10	20	30	40	50	60	80	100
$\lambda = 1$	9.72	19.48	29.34	39.71	49.16	59.49	79.6	99.4
$\lambda = 2$	9.9	19.7	29.9	39.8	49.4	59.6	79.7	99.5
$\lambda = 3$	9.93	19.9	29.92	39.9	49.81	59.9	79.6	99.1
$\lambda = 5$	9.95	19.96	29.93	39.9	49.86	89.89	79.79	99.75

TABLE 4.4 – Nombre moyen d'états atteignables, $\alpha = 3$ et $\beta = 5$.

4.4.4 Un algorithme de rejet pour les APDTR atteignables et complets

Un prolongement naturel de l'étude menée jusqu'ici consiste à envisager un générateur de APDTR accessibles, complets avec exactement n états atteignables.

Un moyen simple de réaliser cet objectif passe par un algorithme à rejet : on engendre des APDTR accessibles et complets à n états jusqu'à obtenir un APDTR dont tous les états sont atteignables. Les résultats des tableaux 4.3 et 4.4 tendent à justifier cette approche pour les paramètres adoptés dans le tableau 4.4. En revanche, les paramètres retenus pour le tableau 4.3 sont plus problématiques. Les résultats des expérimentations menées afin d'évaluer le nombre moyen de rejets est donné dans le tableau 4.5. Dans ce tableau on lit le nombre de tirages nécessaires à la production de 10 APDTR atteignables. Le symbole − signifie qu'après 300 rejets aucun APDTR atteignable n'a été produit. Ces résultats tendent à prouver que l'approche par rejet a du sens pour $\lambda \geqslant 2$ et pour des taille d'alphabets raisonnablement grandes. Pour $\alpha = \beta = 2$, par exemple, l'algorithme à rejets fonctionne pour $\lambda \geqslant 3$. Pour de plus petites valeurs de λ, le nombre d'états doit rester faible.

	nombre d'états →	10	20	30	40	50	60	80	100
	$\lambda = 1$	293.1	-	-	-	-	-	-	-
$\alpha = 2$	$\lambda = 1.5$	24.6	88.8	278.0	-	-	-	-	-
$\beta = 2$	$\lambda = 2$	6.9	20.3	14.9	13.2	17.9	25.2	65.9	95.8
	$\lambda = 3$	1.6	1.5	1.8	2.0	1.9	2.2	2.1	2.1
	$\lambda = 5$	1.1	1.2	1.1	1.2	1.1	1.2	1.2	1.2
	$\lambda = 1$	3.8	5.5	9.4	15.7	38.4	39.3	76.9	76.9
$\alpha = 4$	$\lambda = 1.5$	1.7	2.0	1.7	1.8	1.9	2.0	2.0	1.9
$\beta = 2$	$\lambda = 2$	1.3	1.3	1.31	1.3	1.2	1.2	1.1	1.2
	$\lambda = 3$	1.1	1.1	1.1	1.1	1.0	1.1	1.1	1.1
	$\lambda = 5$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	$\lambda = 1$	2.3	5.5	16.9	23.8	20.1	44.1	80.4	214.2
$\alpha = 2$	$\lambda = 1.5$	1.7	1.8	1.2	2.6	3.0	2.0	1.6	1.9
$\beta = 4$	$\lambda = 2$	1.4	1.2	1.2	1.1	1.3	1.4	1.3	1.1
	$\lambda = 3$	1.0	1.2	1.2	1.0	1.1	1.1	1.0	1.2
	$\lambda = 5$	1.1	1.0	1.0	1.0	1.0	1.0	1.0	1.0

TABLE 4.5 – Nombre moyen de tirages pour obtenir un APDTR atteignable.

4.5 Conclusions et perspectives

Cette partie du travail a mis en œuvre des techniques classiques de combinatoire afin de donner un résultat asymptotique sur la combinatoire de la classe des automates considérés.

Les résultats montrent que le cadre est bien celui décrit dans l'introduction générale : la combinatoire ne permet pas d'envisager des techniques de génération aléatoire basées sur une liste exhaustive des représentants de la classe.

Un cadre général pour le tirage aléatoire uniforme d'automates à pile déterministes a été construit. Les résultats expérimentaux assurent que sous certaines conditions d'acceptation il est possible d'obtenir des automates à pile dont la plupart des états sont atteignables.

On peut noter que la technique employée impose de travailler avec des classes d'automates correspondant à une structure combinatoire adéquate : le choix des classes est guidé par le fait que celle-ci doivent être *spécifiables* dans le sens qui a été donné dans le chapitre 2. La technique d'échantillonnage issue de la méthode symbolique est alors assez directe.

Pour approfondir cette recherche plusieurs pistes sont envisageables :

- Une première idée consiste à trouver une méthode pour engendrer une classe d'automates à pile, *real time*, pour lesquels la plupart des états soient atteignables par pile vide.
- Une autre piste consiste à se libérer de la contrainte *real time*. Ce projet suppose un travail plus approfondi sur l'automate sous-jacent : comportant des ε-transitions, l'automate sous-jacent n'est pas déterministe et on ne peut donc s'appuyer sur les travaux existants.
- Comme pour toute classe, une étude des automates à piles apparaissant dans différents domaines pratique (analyse syntaxique, conversion de requêtes, modélisation, etc.) serait utile pour dégager des tendances et préciser des classes pertinentes pour l'évaluation d'algorithmes.
- La méthode, développée ici, peut s'étendre facilement à toute famille d'automates ayant un automate sous-jacent déterministe et pour lesquels les *sorties* sont générables. On pourrait donc faire des études similaires pour des automates déterministes pondérés, des automates à compteurs,...

Chapitre 5

Génération aléatoire, à isomorphisme près, d'automates non déterministes

Sommaire

5.1 Introduction

Comme exposé dans l'introduction de ce travail, les automates finis jouent un rôle central en théorie des langages formels, ils constituent des outils largement exploités pour aborder des problèmes algorithmiques dans des cadres variés allant du model-checking au traitement de texte.

On rappelle ici que la génération aléatoire de certaines classes d'automates finis a déjà suscité beaucoup d'intérêt [CP05, AMR07, BN07, CN12] particulièrement pour les automates déterministes. On pourra se référer à [Nic14] pour une étude récente sur le sujet.

Le problème de la génération aléatoire d'automates finis non-déterministes pose, par rapport au cas déterministe, deux problèmes spécifiques : d'une part toutes les classes d'isomorphismes n'ont pas le même cardinal et, d'autre part, il convient de savoir générer pour des sous-classes - le cas le plus général n'étant pas nécessairement significatif vis-à-vis des applications.

La difficulté de construction d'un générateur *à isomorphisme près* est essentiellement liée à la taille du groupe d'automorphismes d'un automate non-déterministe à n états. Si le groupe d'automorphismes peut effectivement être trivial, il peut aussi dans certains cas être identifié au groupe des permutations de $[\![1, n]\!]$, il est alors de taille maximale et compte $n!$ éléments. Les conclusions de [Nic14] montrent par ailleurs qu'il peut être important de pouvoir sélectionner certaines classes intéressantes d'automates déterministes. On peut noter par exemple que, pour la loi uniforme, la plupart des automates non-déterministes sur un alphabet Σ, produits dans un échantillon reconnaissent tous les mots de Σ^*. Un générateur qui, moyennant quelques réglages, pourrait produire certaines sous-classes d'automates, se révélerait donc particulièrement intéressant.

On expose dans cette partie comment, à partir de techniques de Monte-Carlo, on peut construire un générateur aléatoire, à isomorphisme près, d'automates non-déterministes. Le générateur initial produit des automates non-déterministes à n états, on l'adapte ensuite pour produire des sous-classes d'automates qui sont susceptibles d'être plus pertinentes pour le problème de l'évaluation d'algorithme. La question de la pertinence des sous-classes engendrées, même si elle présente un grand intérêt, n'est pas traitée en détail et avec toute l'attention qu'elle mériterait dans ce travail. Un des objectifs consiste néanmoins à convaincre de la souplesse des approches qui s'appuient sur les méthodes de type Monte-Carlo. Il s'agit d'ouvrir un champ d'exploration et d'investigation pour d'autres sous-classes non traitées ici en persuadant de l'adaptabilité de notre approche. De plus, les classes pointées (définies ci-après) bornant le degré sortant nous semble déjà assez pertinentes, même si aucun argument autre qu'intuitif nous permet d'étayer ces propos.

Le problème de la génération aléatoire d'automates non-déterministes est exploré dans [TV05] en utilisant des techniques de graphes aléatoires. En revanche, les questions liées à la loi de distribution ou à la question des classes d'isomorphisme n'y est pas abordée. L'utilisation des techniques de chaînes de Markov pour construire des générateurs aléatoires est décrite en détails pour les automates déterministe acycliques et accessibles dans [CF11, CF12].

5.2 Contexte théorique

5.2.1 Isomorphisme d'automates

Isomorphismes, groupe d'isomorphismes

La notion d'isomorphisme d'automates a été définie à la section 1.1.3. On rappelle qu'il s'agit simplement de définir de façon formelle l'égalité, à renommage près des états, de deux automates.

On se limite ici à donner un nouvel exemple avec la figure 5.1 qui présente deux automates isomorphes, avec un isomorphisme φ vérifiant $\varphi(1) = 2$, $\varphi(2) = 1$ et $\varphi(3) = 3$.

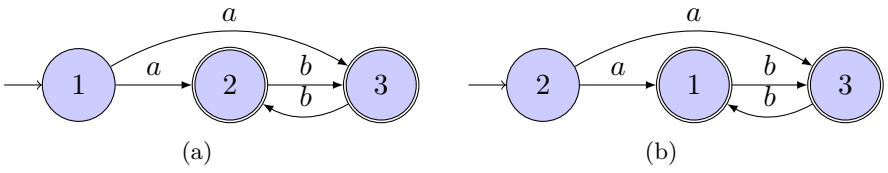


FIGURE 5.1 – Deux automates isomorphes

On rappelle également que le groupe d'automorphismes d'un automate non-déterministe à n états est un sous-groupe du groupe de permutations de $[\![1, n]\!]$ et que son cardinal est

donc un nombre entier naturel diviseur de $n!$. Le groupe d'automorphismes de l'automate représenté 5.2, déjà donné en exemple à la section 1.1.3, admet deux éléments qui sont l'identité de $[\![1,3]\!]$ et la transposition qui échange 2 et 3. Cet exemple sera repris dans la suite de l'exposé.

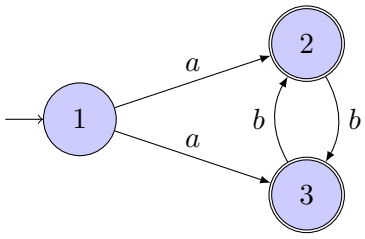


FIGURE 5.2 – Automate $\mathcal{A}$ dont le groupe d'automorphismes, $\mathrm{Aut}(\mathcal{A})$, compte 2 éléments

Classes d'isomorphismes

Il sera essentiel dans la suite de ce travail de bien comprendre les moyens de déterminer la taille du groupe $\mathrm{Aut}(\mathcal{A})$ pour un automate $\mathcal{A}$ donné. Les considérations qui suivent seront des points clefs des sections 5.4 et 5.4.1. Dans ces sections, l'objectif consiste à engendrer uniformément et à isomorphisme près certaines classes d'automates (fermées par isomorphisme).

On considère donc $\mathfrak{C}$ une classe d'automates non-déterministes (fermée par isomorphisme) et n un nombre entier strictement positif. On note $\mathfrak{C}(n)$ l'ensemble des éléments de $\mathfrak{C}$ dont l'ensemble des états est $[\![1,n]\!]$ et γ_n le nombre de classes d'isomorphismes sur $\mathfrak{C}(n)$.

On a donc :

$$\gamma_n = |\mathfrak{C}(n)/\!\sim|$$

où $\sim$ est la relation d'équivalence *"est isomorphe à"*.

Le groupe $\mathfrak{S}_n$ des permutations de $[\![1,n]\!]$ est de cardinal $n!$ et pour tout automate $\mathcal{A}$ de $\mathfrak{C}(n)$ la relation binaire définie sur $\mathfrak{S}_n$ par $\varphi_1(\mathcal{A}) = \varphi_2(\mathcal{A})$ est une relation d'équivalence.

Par ailleurs, pour toutes permutations φ_1 et φ_2 de $[\![1,n]\!]$, on a :

$$(\varphi_1(\mathcal{A}) = \varphi_2(\mathcal{A})) \Leftrightarrow \left(\varphi_2^{-1}\varphi_1(\mathcal{A}) = \mathcal{A}\right) \Leftrightarrow \left(\varphi_2^{-1}\varphi_1 \in \mathrm{Aut}(\mathcal{A})\right)$$

En termes plus mathématiques, il existe une action de groupe naturelle de $\mathfrak{S}_n$ sur $\mathfrak{C}(n)$. Pour tout $\mathcal{A} \in \mathfrak{C}(n)$, $\mathrm{Aut}(\mathcal{A})$ est le sous-groupe stabilisateur de $\mathcal{A}$ pour cette action de groupe, l'orbite de $\mathcal{A}$, c'est-à-dire l'ensemble des automates isomorphes à $\mathcal{A}$ compte donc $\frac{|\mathfrak{S}_n|}{|\mathrm{Aut}(\mathcal{A})|}$ éléments.

En reprenant l'exemple de l'automate de la figure 1.5 repris ici sur la figure 5.3, on a $n = 3$ et :

$$\frac{|\mathfrak{S}_n|}{|\mathrm{Aut}(\mathcal{A})|} = \frac{3!}{2} = 3$$

Chacune des deux colonnes présente l'orbite du même automate. Sur la même ligne sont présentés les deux automates égaux par l'automorphisme non trivial de leur sous-groupe stabilisateur.

Ces considérations conduisent au résultat suivant :

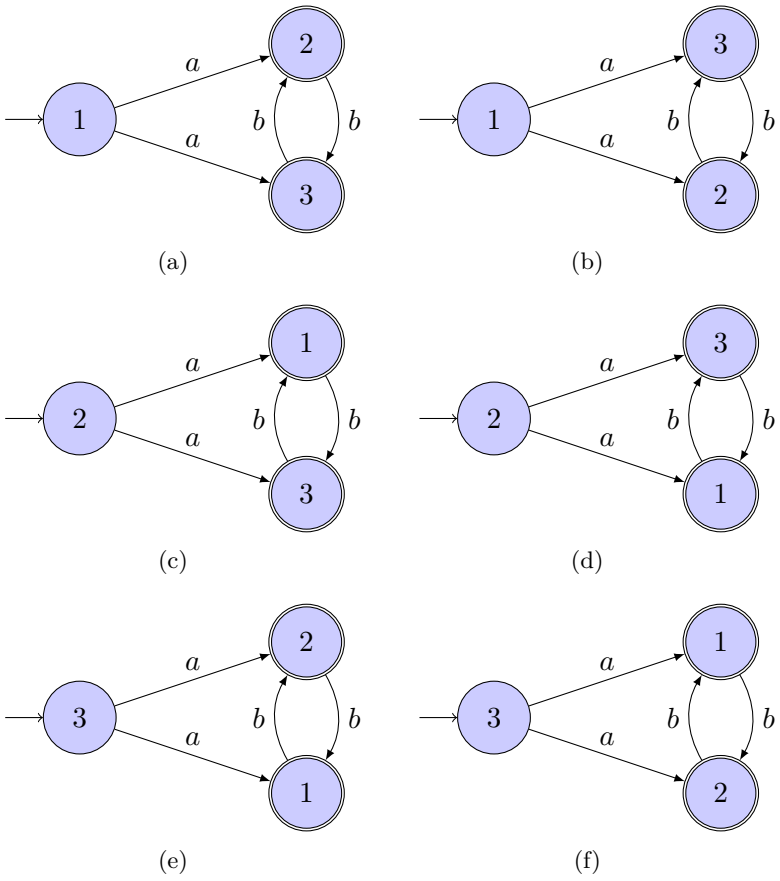


FIGURE 5.3 – Orbite d'un automate : chacune des deux colonnes représente l'orbite du même automate

Proposition 41. *Un générateur aléatoire uniforme des classes d'isomorphismes de* $\mathfrak{C}(n)$ *est obtenu par un tirage aléatoire de chacun des éléments* x *de* $\mathfrak{C}(n)$ *avec la probabilité :*

$$\frac{|\mathrm{Aut}(x)|}{n!\ \gamma_n}.$$

Preuve. Soit H une classe d'isomorphisme de $\mathfrak{C}(n)$. Avec un générateur des éléments x de $\mathfrak{C}(n)$ ayant la propriété de la proposition, la classe H est engendrée avec la probabilité :

$$\sum_{x\in H} \frac{|\mathrm{Aut}(x)|}{n!\ \gamma_n} = \sum_{x\in H} \frac{1}{\gamma_n |H|} = \frac{1}{\gamma_n |H|} \sum_{x\in H} 1 = \frac{|H|}{\gamma_n |H|} = \frac{1}{\gamma_n}.$$

□

Au regard des considérations précédentes, il est intéressant de noter que la taille des orbites d'un automate $\mathcal{A}$ à n états, et par là même la taille de $\mathrm{Aut}(\mathcal{A})$, est susceptible de prendre les valeurs extrêmes : 1 comme $n!$. Les automates à n états $\mathcal{A}_1$ et $\mathcal{A}_2$ décrits dans la suite fournissent deux exemples.

- L'automate $\mathcal{A}_1$ est défini par :

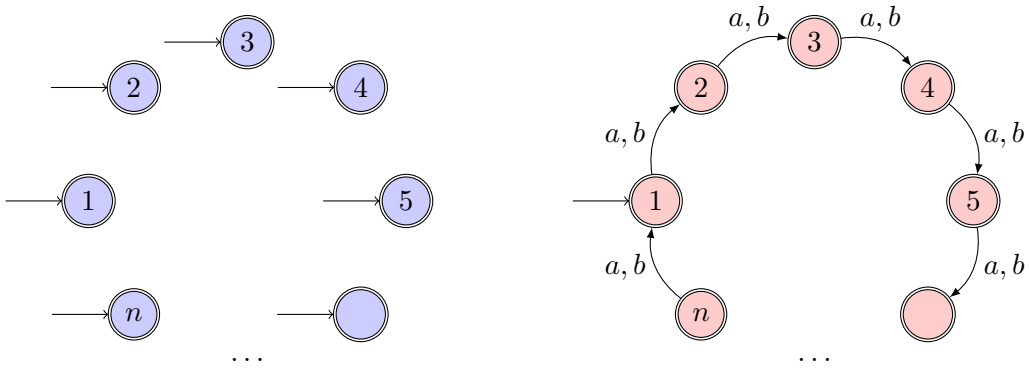
$$\mathcal{A}_1 \stackrel{\text{def}}{=} \{[\![1,n]\!], \Sigma, \emptyset, [\![1,n]\!], [\![1,n]\!]\}\,.$$

  L'orbite de l'automate $\mathcal{A}_1$ est réduite à lui même, on a donc $\mathrm{Aut}(\mathcal{A}_1) = \mathfrak{S}_n$.
- L'automate $\mathcal{A}_2$ défini par :

$$\mathcal{A}_2 \stackrel{\text{def}}{=} \{[\![1,n]\!], \Sigma, \Delta, \{1\}, [\![1,n]\!]\}\,, \quad \Delta \stackrel{\text{def}}{=} \{(i,\alpha,(i+1)[n]),\ \alpha\in\Sigma,\ i\in[\![1,n]\!]\}$$

Tout automate obtenu à partir de $\mathcal{A}_2$ par permutation des états est un automate différent de $\mathcal{A}_2$ qui est dans l'orbite de $\mathcal{A}_2$. L'orbite de $\mathcal{A}_2$ compte donc $n!$ éléments et par là même $\mathrm{Aut}(\mathcal{A}_2) = \{id\}$.

Les automates $\mathcal{A}_1$ et $\mathcal{A}_2$ sont représentés sur la figure 5.4.



Automate $\mathcal{A}_1$ Automate $\mathcal{A}_2$

FIGURE 5.4 – Les automates $\mathcal{A}_1$ et $\mathcal{A}_2$

La proposition 42 fournit un résultat plus général à propos des automates $\mathcal{A}$ pour lesquels $\mathrm{Aut}(\mathcal{A}) = \{id\}$.

Proposition 42 (automorphisme trivial)**.** *On considère $\mathcal{A}$ un automate à n états ; si $\mathcal{A}$ est déterministe et trim, alors* $\mathrm{Aut}(\mathcal{A}) = \{id\}$.

Preuve. On peut supposer que l'ensemble des états de $\mathcal{A}$ est $[\![1, n]\!]$ et que le seul état initial est 1. On considère φ un automorphisme de $\mathcal{A}$. Nécessairement $\varphi(1) = 1$. Si $n \neq 1$, l'ensemble Q_1 des états q tels qu'il existe une lettre a de l'alphabet Σ pour laquelle $(1, a, q)$ est une transition, est non-vide, sinon l'automate ne serait pas trim. Pour tout $q \in Q_1$ on a nécessairement $\varphi(q) = q$. Si $\{1\} \cup Q_1 \neq [\![1, n]\!]$, l'ensemble Q_2 des états p tels qu'il existe une lettre a pour laquelle (q, a, p) est une transition vérifiant $q \in Q_1$, n'est pas vide sinon l'automate ne serait pas trim. Pour tout $p \in Q_2$, on a nécessairement $\varphi(p) = p$. Puisque l'ensemble des états est fini, on peut donc montrer que tous les états sont des points fixes de φ, donc que $\varphi = id$. □

Le problème d'isomorphisme consiste, étant donné deux automates ayant le même nombre d'états, à décider s'ils sont ou non isomorphes. Cette question sera donc essentielle dans la partie consacrée à la génération aléatoire uniforme à isomorphisme près de certaines classes d'automates. Ce point est développé à propos des automates déterministes dans [Boo78]. Il est de même nature que la question du problème d'isomorphisme pour les graphes orientés et le résultat suivant [Luk82] sera utile dans la suite.

Théorème 43. *On considère m un nombre entier strictement positif. Le problème d'isomorphisme pour les graphes orientés de degré inférieur à m est polynomial.*

5.2.2 Différentes classes d'automates

On reprend ici les notations de la section 1.1.1 où $(Q, \Sigma, \delta, I, F)$ désigne un automate $\mathcal{A}$. Pour tout état p de $\mathcal{A}$ et toute lettre a de Σ, on note $p \cdot a$ l'ensemble des états q de Q tels que (p, a, q) soit une transition. Formellement :

$$p \cdot a \stackrel{\text{def}}{=} \{q \in Q,\ (p, a, q) \in \Delta\}\,.$$

Dans la suite de ce chapitre Σ dénote un alphabet fini de cardinal $|\Sigma|$ supérieur ou égal à 2 et m est un nombre entier naturel supérieur ou égal à 2. Tous les automates considérés sont sur cet alphabet.

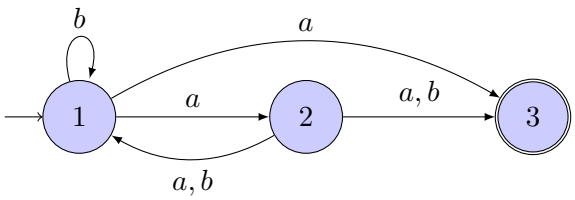
On introduit par ailleurs plusieurs classes d'automates :

- $\mathsf{A}(n)$ est la classe des automates finis pour lesquels l'ensemble Q des états est $[\![1, n]\!]$. Il est clair que tout automate fini à n états peut être identifié à un automate de $\mathsf{A}(n)$.
- $\mathsf{N}(n)$ est la classe des automates finis trims de $\mathsf{A}(n)$.
- $\mathsf{N}_m(n)$ est la classe des automates finis de $\mathsf{N}(n)$ pour lesquels, pour tout état p de Q, il existe au plus m couples (a, q) de $\Sigma \times Q$ tels que (p, a, q) soit une transition.
  $\mathsf{N}_m(n)$ est donc la classe des automates finis trims à n états tels que pour tout $p \in Q$:
  $$\left| \bigcup_{a \in \Sigma} p \cdot a \right| \leqslant m.$$
  Pour cette classe d'automates, il n'y a pas plus de m transitions sortantes par état.
- $\mathsf{N}'_m(n)$ est la classe des automates finis de $\mathsf{N}_m(n)$ pour lesquels, pour tout état p de Q, pour toute lettre a de Σ, il existe au plus m états q tels que (p, a, q) soit une transition.
  $\mathsf{N}'_m(n)$ est donc la classe des automates finis trims à n états tels que pour tout $p \in Q$:
  $$\forall a \in \Sigma, \quad |p \cdot a| \leqslant m.$$
  Pour cette classe d'automates, il n'y a pas plus de m transitions sortantes par état et pour une lettre donnée.

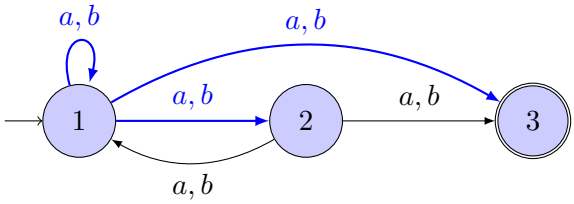
Pour chacune des classes X décrite précédemment, on note $\mathsf{X}^\bullet$ la sous-classe des automates de X pour lesquels l'ensemble des états initiaux est réduit au singleton $\{1\}$.

Au sujet de $\mathsf{X}^\bullet$ on parlera dans la suite de la classe *ponctuée* associée à X.

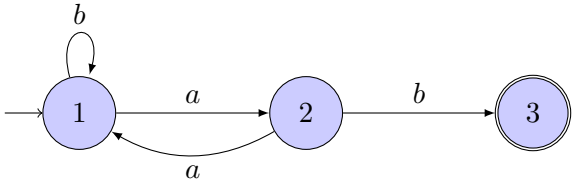
Il est clair que :
$$\mathsf{N}_m(n) \subseteq \mathsf{N}'_m(n) \subseteq \mathsf{N}(n) \subseteq \mathsf{A}(n)$$



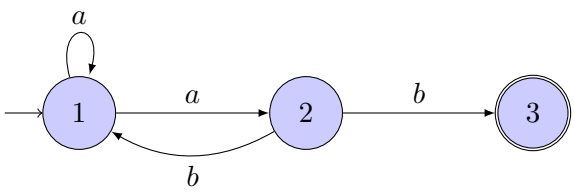
Automate de $\mathsf{N}(3)^\bullet$, $\mathsf{N}_4(3)^\bullet$, $\mathsf{N}'_2(3)^\bullet$



Automate de $\mathsf{N}(3)^\bullet$, $\mathsf{N}_6(3)^\bullet$, $\mathsf{N}'_3(3)^\bullet$



Automate de $\mathsf{N}(3)^\bullet$, $\mathsf{N}_2(3)^\bullet$, $\mathsf{N}'_1(3)^\bullet$



Automate de $\mathsf{N}(3)^\bullet$, $\mathsf{N}_2(3)^\bullet$, $\mathsf{N}'_2(3)^\bullet$

FIGURE 5.5 – Différentes classes d'automates

5.3 Génération aléatoire d'automates finis non-déterministes par chaînes de Markov

5.3.1 Construction des chaînes

Dans cette section sont décrites en détails une famille de chaînes de Markov symétriques et ergodiques associées aux classes $\mathsf{A}(n)$, $\mathsf{N}(n)$, $\mathsf{N}_m(n)$ et $\mathsf{N}'_m(n)$, ainsi qu'aux classes ponctuées associées.

On considère $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ un automate fini.
Nous allons associer à $\mathcal{A}$ des automates qui correspondront aux déplacements possibles dans la chaîne à partir de l'automate $\mathcal{A}$.

Pour tout état q dans Q et tout triplet (p, a, q) élément de $Q \times \Sigma \times Q$, les automates $\mathsf{Ch}_{\mathsf{init}}(\mathcal{A}, q)$, $\mathsf{Ch}_{\mathsf{final}}(\mathcal{A}, q)$ et $\mathsf{Ch}_{\mathsf{trans.}}(\mathcal{A}, (p, a, q))$ sont définis de la façon suivante :

- Si q est un état initial ($q \in I$), alors $\mathsf{Ch}_{\mathsf{init}}(\mathcal{A}, q) \stackrel{\text{def}}{=} (Q, \Sigma, \Delta, I \setminus \{q\}, F)$
  et $\mathsf{Ch}_{\mathsf{init}}(\mathcal{A}, q) \stackrel{\text{def}}{=} (Q, \Sigma, \Delta, I \cup \{q\}, F)$ sinon.
- Si q est un état final ($q \in F$), alors $\mathsf{Ch}_{\mathsf{final}}(\mathcal{A}, q) \stackrel{\text{def}}{=} (Q, \Sigma, \Delta, I, F \setminus \{q\})$
  et $\mathsf{Ch}_{\mathsf{final}}(\mathcal{A}, q) \stackrel{\text{def}}{=} (Q, \Sigma, \Delta, I, F \cup \{q\})$ sinon.
- Si (p, a, q) est une transition ($(p, a, q) \in \Delta$),
  alors $\mathsf{Ch}_{\mathsf{trans.}}(\mathcal{A}, (p, a, q)) \stackrel{\text{def}}{=} (Q, \Sigma, \Delta \setminus \{(p, a, q)\}, I, F)$
  et $\mathsf{Ch}_{\mathsf{trans.}}(\mathcal{A}, (p, a, q)) \stackrel{\text{def}}{=} (Q, \Sigma, \Delta \cup \{(p, a, q)\}, I, F)$ sinon.

Il s'agit aussi de définir la façon de choisir entre les trois types d'automates $\mathsf{Ch}_{\mathsf{init}}(\mathcal{A}, q)$, $\mathsf{Ch}_{\mathsf{final}}(\mathcal{A}, q)$ et $\mathsf{Ch}_{\mathsf{trans.}}(\mathcal{A}, (p, a, q))$.
Pour cela on considère trois nombres réels ρ_1, ρ_2, ρ_3 tels que $0 \leqslant \rho_i \leqslant 1$ et $\rho_1 + \rho_2 + \rho_3 \leqslant 1$.

Reste à définir à partir de ces outils la matrice de transition qui permet de définir les déplacements dans la chaîne.

On considère X une classe d'automates pour lesquels l'ensemble des états est Q. On définit la matrice de transition $S^{\mathsf{X}}_{\rho_1,\rho_2,\rho_3}(x, y)$ sur X de la façon suivante :

- S'il existe un état q tel que $y = \mathsf{Ch}_{\mathsf{init}}(x, q)$, alors $S^{\mathsf{X}}_{\rho_1,\rho_2,\rho_3}(x, y) \stackrel{\text{def}}{=} \frac{\rho_1}{|Q|}$.
- S'il existe un état q tel que $y = \mathsf{Ch}_{\mathsf{final}}(x, q)$, alors $S^{\mathsf{X}}_{\rho_1,\rho_2,\rho_3}(x, y) \stackrel{\text{def}}{=} \frac{\rho_2}{|Q|}$.
- S'il existe un triplet (p, a, q) de $Q \times \Sigma \times Q$ tel que $y = \mathsf{Ch}_{\mathsf{trans.}}(x, (p, a, q))$,
  alors $S^{\mathsf{X}}_{\rho_1,\rho_2,\rho_3}(x, y) \stackrel{\text{def}}{=} \frac{\rho_3}{|\Sigma|.|Q|^2}$.
- Si $y \neq x$ et n'est d'aucune des trois précédentes formes évoquées plus haut,
  alors $S^{\mathsf{X}}_{\rho_1,\rho_2,\rho_3}(x, y) \stackrel{\text{def}}{=} 0$.
- On définit les boucles afin que la matrice $S^{\mathsf{X}}_{\rho_1,\rho_2,\rho_3}(x, y)$ soit stochastique.
  Pour tout automate x, on a $S^{\mathsf{X}}_{\rho_1,\rho_2,\rho_3}(x, x) = 1 - \sum_{y \neq x} S^{\mathsf{X}}_{\rho_1,\rho_2,\rho_3}(x, y)$.

La figure 5.6 illustre comment s'élaborent les mouvements dans la chaîne.

Pour les versions ponctuées des classes on adapte simplement la construction précédente. Pour $\mathsf{X} \in \{\mathsf{N}(n), \mathsf{N}_m(n), \mathsf{N}'_m(n)\}$, et $\rho \in]0, 1[$ on définit la matrice de transition $S^{\mathsf{X}^\bullet}_{\rho}$ sur $\mathsf{X}^\bullet$ par $S^{\mathsf{X}^\bullet}_{\rho} \stackrel{\text{def}}{=} S^{\mathsf{X}}_{0,\rho,1-\rho}$.

Lemme 44. *On considère m, n deux nombres entiers strictement positifs, ρ, ρ_1, ρ_2 et ρ_3, quatre nombres réels strictement positifs tels que $\rho < 1$ et $\rho_1 + \rho_2 + \rho_3 \leqslant 1$. Alors, si $m \geq 2$, $S^{\mathsf{N}(n)}_{\rho_1,\rho_2,\rho_3}$, $S^{\mathsf{N}_m(n)}_{\rho_1,\rho_2,\rho_3}$ et $S^{\mathsf{N}'_m(n)}_{\rho_1,\rho_2,\rho_3}$ comme les versions ponctuées $S^{\mathsf{N}(n)^\bullet}_{\rho}$, $S^{\mathsf{N}_m(n)^\bullet}_{\rho}$ et $S^{\mathsf{N}'_m(n)^\bullet}_{\rho}$ sont irréductibles.*

Preuve. On rappelle que pour les automates considérés : $Q = [\![1, n]\!]$.

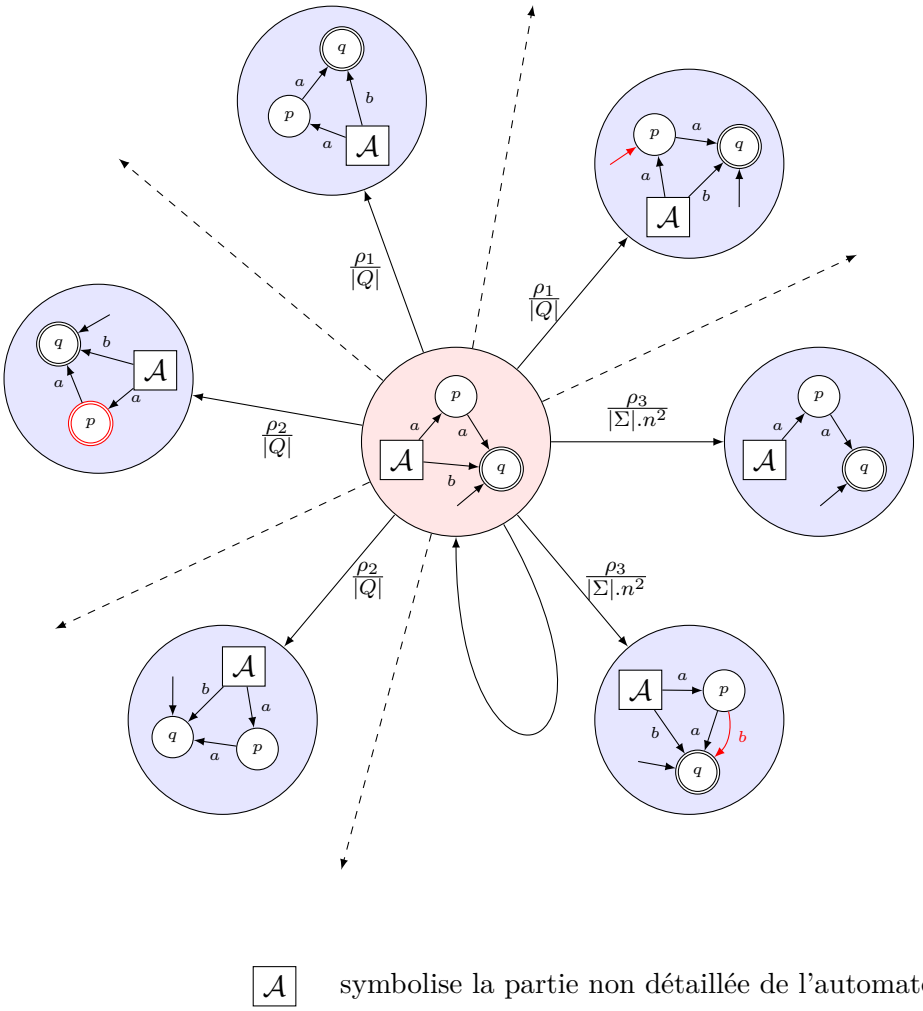


FIGURE 5.6 – Mouvements dans la chaîne. On ne représente au centre en détails qu'une partie de l'automate mis en jeu et 6 déplacements typiques dans la chaîne. D'autres déplacements sont évidemment possibles, symbolisés par les flèches lignes tiretées. La chaîne est construite de façon à ce que l'automate de départ puisse boucler sur lui même.

On considère $\mathsf{X} \in \{\mathsf{N}(n), \mathsf{N}_m(n), \mathsf{N}'_m(n)\}$ et $x \in \mathsf{X}$. On note alors $\mathcal{A}_0$ l'automate $(Q, \Sigma, \emptyset, Q, Q)$. L'automate $\mathcal{A}_0$ est trim par construction et est dans la classe X. Il s'agit maintenant de justifier qu'il existe un chemin dans X de l'automate x à l'automate $\mathcal{A}_0$. Pour cela, on pose $x = (Q, \Sigma, \Delta, I, F)$. Dans l'automate x, faire d'un état, qui ne l'était pas, en état initial ou état final transforme l'automate x en un automate qui reste dans X. Il existe donc un chemin de x à $y = (Q, \Sigma, \Delta, Q, Q)$; ce chemin emprunte des arêtes définies à partir de $\mathsf{Ch}_{\mathsf{init}}$ et $\mathsf{Ch}_{\mathsf{final}}$. À ce stade, puisque tous les états de y sont à la fois initiaux et finals, il existe un chemin de y à $\mathcal{A}_0$, ce chemin emprunte des arêtes qui conduisent à supprimer toutes les transitions de y. Il existe donc un chemin dans X de l'automate x à $\mathcal{A}_0$. Puisque la chaîne de Markov définie est symétrique, il existe aussi un chemin de $\mathcal{A}_0$ à x. En passant par $\mathcal{A}_0$ il est donc possible de définir un chemin entre deux éléments quelconque de X, la chaîne de Markov est donc irréductible.

On considère maintenant $\mathsf{X}^\bullet \in \{\mathsf{N}(n)^\bullet, \mathsf{N}^\bullet_m(n), \mathsf{N}'^\bullet_m(n)\}$ et $x \in \mathsf{X}$. On rappelle que l'ensemble des états initiaux de x est réduit au singleton $\{1\}$. On note a_0 une lettre arbitraire de l'alphabet Σ. On considère alors $\mathcal{A}_1$ l'automate de $\mathsf{N}_m(n)^\bullet$ pour lequel l'ensemble des états finals est Q et dont l'ensemble des transitions est $\{(i, a_0, i+1) \mid 1 \leqslant i < n\}$. L'idée consiste à démontrer, comme dans le cas des versions non ponctuées, qu'il existe un chemin

dans la chaîne de x à $\mathcal{A}_1$:

- En faisant dans l'automate x, d'états qui ne l'étaient pas, des états finals, il est possible de se déplacer dans la chaîne de x à $y = (Q, \Sigma, \Delta, \{1\}, Q)$.
- L'ensemble Δ des transitions peut être considéré comme un graphe orienté dont les sommets sont les états et les arcs les couples (p, q) étiquetés par $a \in \Sigma$, pour tout $(p, a, q) \in \Delta$. Puisque les automates x et y sont accessibles, on peut considérer une partie Δ' de Δ formant un arbre couvrant de Δ enraciné en $\{1\}$. Tous les états de y sont finals, la co-accessibilité est donc préservée par toute modification de l'ensemble des transitions. En supprimant les transitions de $\Delta \backslash \Delta'$ on construit donc un chemin dans la chaîne de y à z défini par $z \stackrel{\text{def}}{=} (Q, \Sigma, \Delta', \{1\}, Q)$.
- Si Δ', vu comme un arbre, admet au moins deux feuilles p et q distinctes, alors les états p et q n'admettent pas dans z de transition sortante. En ajoutant la transition (p, a_0, q) à l'automate celui-ci reste dans $\mathsf{X}^\bullet$. On supprime alors l'unique transition arrivant en q. L'automate obtenu est encore dans $\mathsf{X}^\bullet$. À la suite de ces deux mouvements dans la chaîne, on parvient à un automate de la forme $(Q, \Sigma, \Delta'', \{1\}, Q)$ où Δ'' peut être vu comme un arbre ayant strictement moins de feuilles que Δ'. En répétant inductivement ce procédé, on peut donc construire un chemin de z à un automate u de la forme :

  $$u = (\llbracket 1, n \rrbracket, \Sigma, \{(\varphi(i), a_i, \varphi(i+1)) \mid 1 \leqslant i < n\}, \{1\}, \llbracket 1, n \rrbracket)$$

  où φ est une permutation de Q qui fixe 1.
- Puisque $m \geqslant 2$, on peut éventuellement, pour chaque état de u, ajouter une transition sortante. Pour tout $i \in \llbracket 1, n-1 \rrbracket$, si $(i, a_0, i+1)$ n'appartient pas déjà à l'ensemble des transition, il est possible de l'ajouter. La fin du procédé consiste alors à supprimer toutes les transitions qui ne sont pas de la forme $(i, a_0, i+1)$, on obtient ainsi un chemin de u à $\mathcal{A}_1$.

On a donc montré qu'il existait dans la chaîne un chemin de x à $\mathcal{A}_1$.

Puisque la chaîne est symétrique on peut donc conclure qu'elle est également irréductible. □

Lemme 45. *On considère m, n deux nombres entiers strictement positifs tels que $m \geq 2$, ρ, ρ_1, ρ_2 et ρ_3, quatre nombres réels strictement positifs tels que $\rho < 1$ et $\rho_1 + \rho_2 + \rho_3 \leqslant 1$, alors $S^{\mathsf{N}(n)}_{\rho_1,\rho_2,\rho_3}$, $S^{\mathsf{N}_m(n)}_{\rho_1,\rho_2,\rho_3}$ et $S^{\mathsf{N}'_m(n)}_{\rho_1,\rho_2,\rho_3}$ comme les versions ponctuées $S^{\mathsf{N}(n)^\bullet}_{\rho}$, $S^{\mathsf{N}_m(n)^\bullet}_{\rho}$ et $S^{\mathsf{N}'_m(n)^\bullet}_{\rho}$ sont apériodiques.*

Preuve. On considère $\mathsf{X} \in \{\mathsf{N}(n), \mathsf{N}_m(n), \mathsf{N}'_m(n)\}$ et $x \in \mathsf{X}$. En reprenant les notations du lemme 44, on peut dire qu'il existe un chemin de longueur n_x de x à $\mathcal{A}_0$. Il existe donc un cycle de longueur $2n_x$ qui passe par x.

Par ailleurs, puisqu'en particulier 1 est un état initial dans $\mathcal{A}_0$, l'état 1 n'est plus accessible dans l'automate $\mathsf{Ch}_{\mathsf{init}}(\mathcal{A}_0, 1)$ qui par là même n'appartient pas à la classe X. En conséquence $S^{\mathsf{X}}(\mathcal{A}_0, \mathcal{A}_0) \neq 0$ et en empruntant cette boucle, on peut donc considérer un cycle de longueur $2n_x + 1$ qui passe par x. Finalement, puisque $2n_x \wedge 2n_x + 1 = 1$, la chaîne est apériodique.

On considère maintenant $\mathsf{X}^\bullet \in \{\mathsf{N}(n)^\bullet, \mathsf{N}^\bullet_m(n), \mathsf{N}'^\bullet_m(n)\}$ et $x \in \mathsf{X}$. En reprenant à nouveau les notations du lemme 44, on peut dire que depuis $\mathcal{A}_1$, si on tire la transition $(1, a_0, 2)$ (ce qui est possible puisque la probabilité d'obtenir cette transition est $\frac{(1-\rho)}{|\Sigma| n^2}$, donc différente de 0), on réalise alors une boucle de $\mathcal{A}_1$ vers $\mathcal{A}_1$. En utilisant le même argument que dans la démonstration proposée pour les versions non ponctuées des classes, on conclut que la chaîne de Markov est apériodique. □

Proposition 46. *On considère m, n deux nombres entiers strictement positifs tels que $m \geq 2$, ρ, ρ_1, ρ_2 et ρ_3, quatre nombres réels strictement positifs tels que $\rho < 1$ et $\rho_1 + \rho_2 + \rho_3 \leqslant 1$. Alors les chaînes de Markov de matrices $S^{\mathsf{N}(n)}_{\rho_1,\rho_2,\rho_3}$, $S^{\mathsf{N}_m(n)}_{\rho_1,\rho_2,\rho_3}$, $S^{\mathsf{N}'_m(n)}_{\rho_1,\rho_2,\rho_3}$, $S^{\mathsf{N}(n)^\bullet}_{\rho}$, $S^{\mathsf{N}_m(n)^\bullet}_{\rho}$ et $S^{\mathsf{N}'_m(n)^\bullet}_{\rho}$ sont ergodiques et et leur distribution stationnaire et la distribution uniforme.*

Preuve. Les lemmes 45 et 44 assurent que les chaînes sont ergodiques. Les différentes matrices considérées sont par ailleurs symétriques, la distribution stationnaire de chacune de ces chaînes est donc la distribution uniforme (sur chacune des classes d'automates associés). □

Dans la pratique le calcul de X_{t+1} à partir de X_t est réalisé de la manière suivante :

- On choisit d'abord avec les probabilités ρ_1, ρ_2 et ρ_3 si on modifie un état initial, un état final ou une transition.
- Dans tous les cas, toutes les changements possibles sont réalisés avec la même probabilité.
- Si l'automate obtenu appartient à la classe ciblée, X_{t+1} prend cette valeur sinon $X_{t+1} = X_t$.
  Puisque la vérification de l'appartenance de l'automate à la classe souhaitée ($\mathsf{N}(n)$, $\mathsf{N}_m(n)$ ou $\mathsf{N}'_m(n)$) se fait en un temps polynomial en n, le calcul de X_{t+1} à partir de X_t est réalisé en un temps polynomial en n.

On définit la chaîne de Markov paresseuse sur $\mathsf{A}(n)$ par :

- $L^{\mathsf{A}(n)}_{\rho_1,\rho_2,\rho_3}(x,y) \stackrel{\text{def}}{=} \frac{1}{2} S^{\mathsf{A}(n)}_{\rho_1,\rho_2,\rho_3}(x,y)$ si $x \neq y$
- et $L^{\mathsf{A}(n)}_{\rho_1,\rho_2,\rho_3}(x,x) \stackrel{\text{def}}{=} \frac{1}{2} + \frac{1}{2} S^{\mathsf{A}(n)}_{\rho_1,\rho_2,\rho_3}(x,x)$ sinon.

Il est établi qu'une chaîne de Markov symétrique et sa version paresseuse ont un temps de mélange similaire.

5.3.2 Temps de mélange dans la chaîne

On réussit à établir un résultat intéressant à propos du temps de mélange dans la chaîne grâce à la proposition 47.

Proposition 47. *L'ε-temps de mélange $\tau(\varepsilon)$ de $L^{\mathsf{A}(n)}_{\rho_1,\rho_2,\rho_3}$ est majoré par :*

$$\max\left(\left\lceil \frac{n}{\rho_1}\left(\log(n) + \log\left(\frac{1}{\rho_1\varepsilon}\right)\right)\right\rceil, \left\lceil \frac{n}{\rho_2}\left(\log(n) + \log\left(\frac{1}{\rho_2\varepsilon}\right)\right)\right\rceil, \left\lceil \frac{|\Sigma|n^2}{\rho_3}\left(\log(|\Sigma|n^2) + \log\left(\frac{1}{\rho_3\varepsilon}\right)\right)\right\rceil\right)$$

Preuve. La preuve du résultat de la proposition 47 s'appuie sur des résultats classiques de la théorie des chaînes de Markov. Les outils mis en jeu comme le *couplage* et la notion de *temps stationnaire* sont décrits dans la section 3.1.3.

Dans le cas qui nous occupe, il suffit de remarquer que tout automate non-déterministe peut être codé par des ensembles d'applications booléennes et que les déplacements dans la chaîne sont proches de la marche aléatoire paresseuse sur l'hypercube.

À tout automate non-déterministe $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ à n états on peut associer une application φ_I de l'ensemble de ses états dans l'ensemble $\{0,1\}$ par :

$$\forall q \in Q,\ \varphi_I(q) = \begin{cases} 1 \text{ si } q \in I \\ 0 \text{ sinon} \end{cases}$$

l'application φ_I est donc l'application caractéristique $\mathbb{1}_I$, du sous ensemble I de Q.
De la même façon, on définit les applications $\varphi_F : Q \to \{0,1\}$ et $\varphi_\Delta : Q \times \Sigma \times Q$ comme

les fonctions caractéristiques $\mathbb{1}_F$ et $\mathbb{1}_\Delta$ des ensembles des états finals et de l'ensemble des transitions respectivement considérés comme sous-ensembles de Q et $Q \times \Sigma \times Q$.

L'automate $\mathcal{A}$ est entièrement défini par les trois applications φ_I, φ_F et φ_Δ.
La chaîne de Markov peut donc être décomposée en trois marches aléatoires, les deux premières sur l'hypercube de dimension n (pour les états initiaux et les états finals), la troisième sur l'hypercube de dimension $n^2|\Sigma|$ (pour les transitions).

À chaque pas on se déplace avec la probabilité ρ_1 dans le premier hypercube, avec la probabilité ρ_2 dans le second hypercube et avec la probabilité $\rho_3 = 1 - \rho_1 - \rho_2$ dans le troisième hypercube.

On procède alors comme dans la démonstration de l'inégalité 3.11 de la page 52. Avec les mêmes notations, on a pour tout nombre entier naturel N non-nul :

$$\mathbb{P}(\tau > N) = \mathbb{P}\left(\bigcup_{i=1}^{n} I_i \cup \bigcup_{j=1}^{n} F_j \cup \bigcup_{k=1}^{|\Sigma|n^2} \Delta_k\right)$$

où I_i, F_j et Δ_k sont les événements suivants :

- I_i : le i-ème état n'est pas obtenu lors des N tirages dans l'hypercube de dimension n, H_I, associé au choix des états initiaux.
- F_j : le j-ème état n'est pas obtenu lors des N tirages dans l'hypercube de dimension n, H_F, associé au choix des états finals.
- Δ_k : la k-ème transition n'est pas obtenue lors des N tirages dans l'hypercube de dimension $|\Sigma|n^2$, H_Δ, associé au choix des transitions.

En particulier :

$$\mathbb{P}(\tau > N) \leqslant \sum_{i=1}^{n} \mathbb{P}(I_i) + \sum_{j=1}^{n} \mathbb{P}(F_j) + \sum_{k=1}^{|\Sigma|n^2} \mathbb{P}(\Delta_k)$$

Or :

$$\mathbb{P}(I_i) = \left(\rho_1\left(1 - \frac{1}{n}\right) + (1 - \rho_1)\right)^N = \left(1 - \frac{\rho_1}{n}\right)^N$$

En effet, si on emprunte avec la probabilité ρ_1 l'hypercube H_I, le i-ème état n'est pas obtenu avec la probabilité $\left(1 - \frac{1}{n}\right)$. Avec la probabilité $1 - \rho_1$, le tirage ne se fait pas dans l'hypercube H_I. On établit de la même manière que :

$$\mathbb{P}(F_j) = \left(1 - \frac{\rho_2}{n}\right)^N, \quad \mathbb{P}(\Delta_k) = \left(1 - \frac{\rho_3}{|\Sigma|n^2}\right)^N$$

On a donc :

$$\mathbb{P}(\tau > N) \leqslant n\left(1 - \frac{\rho_1}{n}\right)^N + n\left(1 - \frac{\rho_2}{n}\right)^N + |\Sigma|n^2\left(1 - \frac{\rho_3}{|\Sigma|n^2}\right)^N$$

Les applications $N \mapsto \left(1 - \frac{\rho_1}{n}\right)^N$, $N \mapsto \left(1 - \frac{\rho_2}{n}\right)^N$ et $N \mapsto \left(1 - \frac{\rho_3}{|\Sigma|n^2}\right)^N$ sont décroissantes. Or, pour tout nombre réel strictement positif ρ et pour tout nombre entier N tel que $N \geqslant \frac{n}{\rho}\left(\log(n) + \log\left(\frac{1}{\rho\varepsilon}\right)\right)$, on a :

$$n\left(1 - \frac{\rho}{n}\right)^N \leqslant n\left(1 - \frac{\rho}{n}\right)^{\frac{n}{\rho}\left(\log(n) + \log\left(\frac{1}{\rho\varepsilon}\right)\right)} \leqslant \underbrace{n\exp\left(-\log(n) + \log(\rho\varepsilon)\right)}_{=\rho\varepsilon}$$

on démontre de la même façon que :

$$\left(N \geqslant \frac{|\Sigma|n^2}{\rho_3}\left(\log(|\Sigma|n^2) + \log\left(\frac{1}{\rho_3\varepsilon}\right)\right)\right) \Rightarrow \left(|\Sigma|n^2\left(1 - \frac{\rho_3}{|\Sigma|n^2}\right)^N \leqslant \rho_3\varepsilon\right)$$

$n \to$	5	10	15	20
$\mathsf{A}(n)$, 2 lettres	0.94	0.95	0.99	1.0
$\mathsf{A}(n)$, 4 lettres	0.95	0.96	1.0	1.0

TABLE 5.1 – Proportion d'automates *trim* rencontrés.

Pour tout nombre entier N tel que :

$$N \geqslant \max\left(\frac{n}{\rho_1}\left(\log(n)+\log\left(\frac{1}{\rho_1\varepsilon}\right)\right), \frac{n}{\rho_2}\left(\log(n)+\log\left(\frac{1}{\rho_2\varepsilon}\right)\right), \frac{|\Sigma|n^2}{\rho_3}\left(\log(|\Sigma|n^2)+\log\left(\frac{1}{\rho_3\varepsilon}\right)\right)\right)$$

on a donc :

$$\mathbb{P}\left(\tau > N\right) \leqslant \varepsilon$$

Le résultat de la proposition se déduit immédiatement de ces inégalités (en s'appuyant sur le résultat du théorème 27 et l'inégalité 3.11 de la page 52).

□

À ce stade de notre travail, nous ne sommes pas encore capables de donner en encadrement des temps de mélange des chaînes de Markov construites. Des expériences pratiques avec des tailles d'alphabets variées, tendent à montrer qu'une part importante des automates engendrés par les chaînes paresseuses (avec un mélange en n^3) sont trim. Les résultats expérimentaux sont montrés dans la table 5.1. Il est clair cependant qu'il ne s'agit là que d'une estimation qui demande un étayage théorique plus solide, notamment si on s'intéressait aux classes ponctuées.

5.4 Génération aléatoire d'automates non-déterministes à isomorphisme près

Cette section montre comment mettre en œuvre l'algorithme de Metropolis-Hastings pour construire un générateur uniforme d'automates non-déterministes, à isomorphisme près. Pour atteindre cet objectif, le point clef consiste *simplement* à calculer les tailles des groupes d'automorphismes des automates impliqués. Dans la section 5.4.3 on démontre que ce calcul est polynomialement équivalent au problème d'isomorphisme des automates impliqués. Pour les classes $\mathsf{N}_m(n)$, $\mathsf{N}_m(n)^\bullet$, $\mathsf{N}'_m(n)$ et $\mathsf{N}'^\bullet_m$, on montre plus précisément que ce calcul peut être fait dans un temps polynomial en n (le cardinal de l'ensemble Q des états) pour m fixé. La section 5.4.5 montre comment calculer en pratique les tailles des groupes d'automorphismes en utilisant des techniques d'étiquetage. Finalement, les résultats expérimentaux sont donnés dans la section 5.4.6.

5.4.1 L'algorithme de Metropolis-Hastings

L'idée consiste à appliquer l'algorithme de Metropolis-Hastings en utilisant, suivant les cas, une des matrices $S(x,y)$ de la chaîne non-modifiée. Les différentes matrices S sont symétriques, on est donc dans le cas de la formule 3.16 de la section 3.1.5. Dans ces formules, la distribution π ciblée est définie par :

$$\forall x \in, \quad \pi(x) \stackrel{\text{def}}{=} \frac{|\mathrm{Aut}(x)|}{n!\,\gamma_n}$$

il n'est donc pas nécessaire de connaître γ_n pour obtenir la nouvelle matrice de transition P. En effet, on a :

$$\forall (x,y) \in \Omega^2, \quad P(x,y) \stackrel{\text{def}}{=} \begin{cases} S(x,y)\left[1 \wedge \frac{|\mathrm{Aut}(y)|}{|\mathrm{Aut}(x)|}\right] & \text{si } x \neq y \\ 1 - \sum_{z \in \Omega \setminus \{x\}} S(x,z)\left[1 \wedge \frac{|\mathrm{Aut}(z)|}{|\mathrm{Aut}(x)|}\right] & \text{si } x = y \end{cases}$$

Une application directe de l'algorithme de Metropolis-Hastings suppose, pour définir les mouvements possibles à partir d'un automate x, de calculer pour chacun des voisins de x dans le graphe associé à la chaîne, la taille de leur groupe d'automorphismes.

Puisqu'un automate à n états admet un nombre de voisins de l'ordre de $|\Sigma|n^2$, cette entreprise peut supposer des calculs assez considérables pour chaque déplacement. Néanmoins les évaluations pratiques montrent que les groupes d'automorphismes d'un automate sont de taille assez modeste. Ainsi, l'approche exposée dans la section 3.1.5 peut-elle être envisagée de façon raisonnable. On observe en pratique un très petit nombre de rejets.

La question du calcul de la taille des groupes d'automorphismes associés aux automates non-déterministes est traitée dans la section 5.4.3.

5.4.2 Une méthode alternative

En supposant que le calcul de la taille des groupes d'automorphismes associés aux automates non-déterministes puisse être réalisé en un temps raisonnable, on peut envisager de s'appuyer sur un algorithme d'acceptation-rejet tel que décrit dans la section 3.1.4 de la page 53.

Ici, pour chaque automate $\mathcal{A}$ de $\mathfrak{C}_n$, la probabilité cible est celle de la proposition 41 :

$$\frac{|\mathrm{Aut}(\mathcal{A})|}{|\gamma_n|\, n!}$$

Le générateur dont on dispose simule une distribution uniforme pour laquelle chaque automate $\mathcal{A}$ est tiré avec une probabilité $\frac{1}{|\mathfrak{C}_n|}$. La constante c de la théorie exposée dans la section 3.1.4 est donc ici définie par :

$$c \stackrel{\text{def}}{=} \max_{\mathcal{A} \in \mathfrak{C}_n} \frac{|\mathfrak{C}_n|}{|\gamma_n|} \frac{|\mathrm{Aut}(\mathcal{A})|}{n!} = \frac{|\mathfrak{C}_n|}{|\gamma_n|} \max_{\mathcal{A} \in \mathfrak{C}_n} \frac{|\mathrm{Aut}(\mathcal{A})|}{n!}$$

Le nombre moyen de rejets est donc égal à la taille moyenne des classes d'isomorphisme multiplié par l'inverse de la taille de la plus petite des classes.

On a vu sur les exemples de la figure 5.4 qu'il est possible d'obtenir un automate pour lequel $|\mathrm{Aut}(\mathcal{A})| = n!$. Même si dans ce cas l'automate n'est pas *trim*, on conçoit qu'il est possible d'obtenir, suivant la classe considérée, une valeur de $\max_{\mathcal{A} \in \mathfrak{C}_n} \frac{|\mathrm{Aut}(\mathcal{A})|}{n!}$ égale ou proche de 1. Ce qui est déterminant dans l'efficacité de cette méthode de rejet c'est la taille moyenne $\frac{|\mathfrak{C}_n|}{|\gamma_n|}$ des classes d'isomorphismes. Si cette taille moyenne est faible, c'est-à-dire si la taille moyenne des groupes d'automorphismes est grande, la méthode de rejet peut être intéressante.

Dans la pratique, la taille des groupes automorphismes est modeste et la taille des classes d'isomorphismes importante. Cette analyse tend à pencher pour la méthode s'appuyant sur l'algorithme de Metropolis-Hastings développée dans la section précédente. Le tableau 5.2 montre quelques résultats expérimentaux illustrant ces propos : la seconde colonne donne la taille moyenne des groupes d'automorphisme, la seconde la valeur maximale observée et la troisième le nombre de fois où cette valeur maximale est observée.

Classe	moyenne	taille maximale observée	nb max
$\mathsf{N}_2(5)$	1.023	6	1
$\mathsf{N}_2(8)$	1.012	6	1
$\mathsf{N}_2(10)$	1.015	2	15
$\mathsf{N}_2(15)$	1.007	2	7
$\mathsf{N}_2(20)$	1.001	2	1
$\mathsf{N}_3(5)$	1.031	6	2
$\mathsf{N}_3(8)$	1.015	2	15
$\mathsf{N}_3(10)$	1.015	2	18
$\mathsf{N}_3(15)$	1.005	2	5
$\mathsf{N}_3(20)$	1	1	1000
$\mathsf{N}(5)$	1.022	2	22
$\mathsf{N}(8)$	1.01	2	10
$\mathsf{N}(10)$	1.018	2	17
$\mathsf{N}(15)$	1.005	2	23
$\mathsf{N}(20)$	1.002	2	2

TABLE 5.2 – Tailles mesurées des groupes d'automorphismes, en utilisant un temps de mélange de n^3, et 1000 tests à chaque fois, pour un alphabet de taille 2.

5.4.3 Taille du groupe d'automorphismes

Cette section est consacrée aux techniques mises en œuvre pour déterminer $|\mathrm{Aut}(\mathcal{A})|$ le cardinal du groupe des automorphismes d'un automate non-déterministe donné. La méthode consiste à un nombre d'appels polynomial du problème d'isomorphisme. Il s'agit d'une adaptation du résultat similaire concernant les graphes orientés [Mat79].

On considère un automate non-déterministe $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, Q' une partie de l'ensemble Q des états, σ une application bijective de Q' dans $[\![1, |Q'|]\!]$ (σ est une fonction qui numérote les éléments de Q'), a_0 une lettre de l'alphabet Σ, fixée arbitrairement, et enfin ℓ le nombre entier défini par :

$$\ell \stackrel{\text{def}}{=} |Q| + |Q'| + 2$$

Pour tout état $r \in Q \backslash Q'$ on note $\mathcal{A}_r^{Q'}$ l'automate $(Q_r, \Sigma, \Delta_r, I, F)$ où

$$Q_r \stackrel{\text{def}}{=} Q \cup \{(p, i) \mid p \in Q \text{ et } 1 \leqslant i \leqslant \ell\}$$

et :

$$\begin{aligned}\Delta_r \stackrel{\text{def}}{=}\ & \Delta \cup \{(p, a_0, (p, 1)) \mid p \in Q\} \\ & \cup \{((p, i), a_0, (p, i+1)) \mid p \in Q' \text{ et } 1 \leqslant i < |Q| + 1 + \sigma(p)\} \\ & \cup \{((r, i), a_0, (r, i+1)) \mid 1 \leqslant\ i < \ell\} \\ & \cup \{((p, i), a_0, (p, i+1)) \mid p \notin Q' \cup \{r\} \text{ et } 1 \leqslant i < |Q| + 1\}\end{aligned}$$

Pour tout état p de Q on appelle *terminaison* de p et on note π_p, le plus long chemin dans $\mathcal{A}_r^{Q'}$ qui relie p à un état de la forme (p, i) par des transitions étiquetées par a_0. Plus formellement :

- Si $p = r$, alors :

$$\pi_r = (r, a_0, (r, 1))((r, 1), a_0, (r, 2)) \dots ((r, \ell - 1), a_0, (r, \ell))$$

- Si $p \in Q'$:

$$\pi_p = (p, a_0, (p, 1))((p, 1), a_0, (p, 2)) \dots ((p, |Q| + \sigma(p)), a_0, (p, |Q| + \sigma(p) + 1))$$

- Si $p \notin Q' \cup \{r\}$:

$$\pi_p = (p, a_0, (p,1))((p,1), a_0, (p,2)) \dots ((p,|Q|), a_0, (p,|Q|+1))$$

La figure 5.7 représente un automate $\mathcal{A}$ et l'automate $\mathcal{A}_r^{Q'}$ associé.

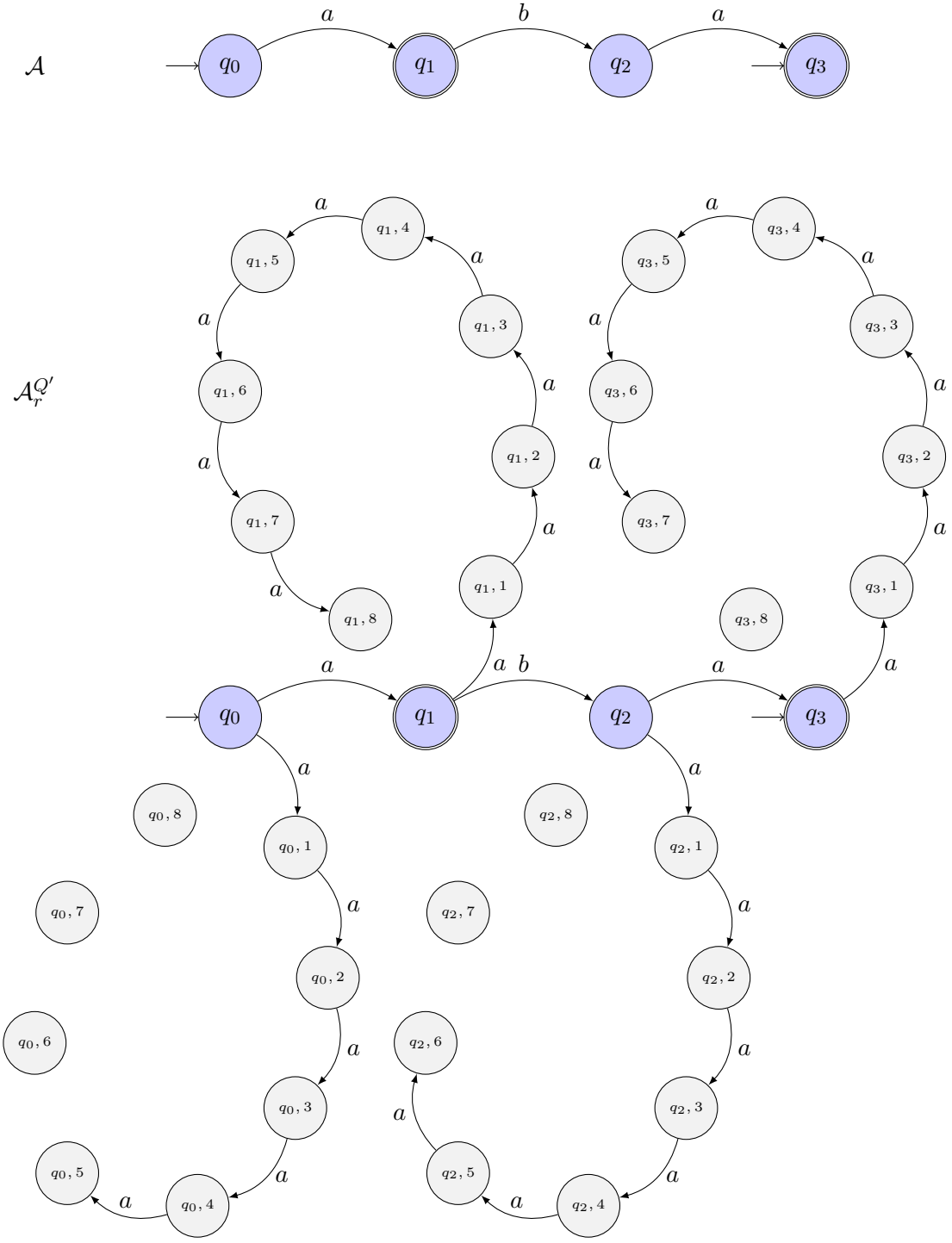


FIGURE 5.7 – Exemple de graphe $\mathcal{A}_r^{Q'}$ associé à un automate $\mathcal{A}$, $a_0 = a$, $r = q_1$, $Q' = \{q_2, q_3\}$, $\sigma(q_2) = 1$, $\sigma(q_3) = 2$

On note que la taille de $\mathcal{A}_r^{Q'}$ est polynomiale en la taille de $\mathcal{A}$.

Les deux lemmes qui suivent donnent les arguments théoriques qui permettent de réduire le problème de la combinatoire des groupes d'automorphismes à celui du problème d'isomorphisme.

Lemme 48. *On considère $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ un automate non-déterministe, Q' une partie non-vide de l'ensemble Q des états de $\mathcal{A}$. Pour tous $q, q' \in Q \backslash Q'$, il existe $\varphi \in \mathrm{Aut}_{Q'}(\mathcal{A})$ tel que $\varphi(q) = q'$ si et seulement si $\mathcal{A}_q^{Q'}$ et $\mathcal{A}_{q'}^{Q'}$ sont isomorphes.*

Preuve. Dans $\mathcal{A}_r^{Q'}$, pour tout état $p \in Q' \cup \{r\}$, on note π_p le chemin défini par

$$\pi_p \stackrel{\text{def}}{=} (p, a_0, (p,1))((p,1), a_0, (p,2)) \dots ((p, \ell' - 1), a_0, (p, \ell'))$$

appelée la *terminaison* de l'état p.

Supposons qu'il existe $\varphi \in \mathrm{Aut}_{Q'}(\mathcal{A})$ tel que $\varphi(q) = q'$.

Soit alors $\hat{\varphi}$ l'application de l'ensemble des états de $\mathcal{A}_q^{Q'}$ sur l'ensemble des états de $\mathcal{A}_{q'}^{Q'}$ définie par, si $p \in Q$, $\hat{\varphi}(p) \stackrel{\text{def}}{=} \varphi(p)$ et si (p,i) est un état de $\mathcal{A}_q^{Q'}$, $\hat{\varphi}((p,i)) \stackrel{\text{def}}{=} (\varphi(p), i)$. L'application $\hat{\varphi}$ est bien définie puisque p et $\varphi(p)$ ont des terminaisons de même longueur. Par construction $\hat{\varphi}$ est un isomorphisme.

Supposons réciproquement qu'il existe un isomorphisme Φ de $\mathcal{A}_q^{Q'}$ sur $\mathcal{A}_{q'}^{Q'}$.

Puisque les isomorphismes conservent l'accessibilité et la co-accessibilité des états et puisque $\mathcal{A}$ est trim, p est un élément de Q si et seulement si $\Phi(p) \in Q$. Soit ϕ la restriction de Φ à Q. Puisque Φ est un morphisme, ϕ est par construction un automorphisme de $\mathcal{A}$. Par ailleurs, puisque Φ conserve la longueur des terminaisons, pour tout $p \in Q'$, $\Phi(p) = p$. On justifie finalement par le même type d'arguments que $\Phi(q) = q'$, ce qui termine la démonstration du lemme. □

Lemme 49. *On considère $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ un automate non-déterministe, Q' une partie non-vide de l'ensemble Q des états de $\mathcal{A}$.*
Pour tout $q \in Q'$, il existe un nombre entier d tel que :

$$|\mathrm{Aut}_{Q' \backslash \{q\}}(\mathcal{A})| = d|\mathrm{Aut}_{Q'}(\mathcal{A})|.$$

Par ailleurs, d peut être calculé sur la base d'un nombre polynomial de tests sur les automates de la forme $\mathcal{A}_r^{Q' \backslash \{q\}}$.

Preuve. On pose $d \stackrel{\text{def}}{=} |\{\phi(q) \mid \phi \in \mathrm{Aut}_{Q' \backslash \{q\}}(\mathcal{A})\}|$ et on considère la relation $\sim_q$ définie sur $\mathrm{Aut}_{Q' \backslash \{q\}}(\mathcal{A})$ Par :

$$(\phi_1 \sim_q \phi_2) \Leftrightarrow (\phi_1(q) = \phi_2(q)).$$

On a alors $\phi_1 \sim_q \phi_2$ si et seulement si $\phi_1 \phi_2^{-1} \in \mathrm{Aut}_{Q'}(\mathcal{A})$. La relation $\sim_q$ est donc une congruence et par conséquent :

$$|\mathrm{Aut}_{Q' \backslash \{q\}}(\mathcal{A})| = d|\mathrm{Aut}_{Q'}(\mathcal{A})|.$$

Pour calculer d, il suffit de déterminer quels éléments de $Q \setminus Q'$ appartiennent à $\{\phi(q) \mid \phi \in \mathrm{Aut}_{Q' \backslash \{q\}}(\mathcal{A})\}$. Cela revient, pour $p \in Q \setminus Q'$, à chercher s'il existe un automorphisme ϕ de $\mathrm{Aut}_{Q' \backslash \{q\}}(\mathcal{A})$ tel que $\phi(q) = p$. Le résultat du lemme 48 assure que c'est possible en temps polynomial. □

Le lemme 49 fournit une méthode pour calculer la taille des groupes d'automorphismes en testant si deux automates non-déterministes sont isomorphes. Plus précisément, puisque $\mathrm{Aut}_Q(\mathcal{A})$ est réduit à l'identité et que par ailleurs $\mathrm{Aut}(\mathcal{A}) = \mathrm{Aut}_\emptyset(\mathcal{A})$, on peut montrer par une induction s'appuyant sur le résultat du lemme 49, que $\mathrm{Aut}(\mathcal{A}) = d_1 \dots d_{|Q|}$, où chaque d_i peut être calculé par un nombre polynomial de tests d'isomorphismes. C'est ainsi que la question du calcul du cardinal des groupes stabilisateurs se réduit au test d'isomorphisme de deux automates.

5.4.4 Le problème d'isomorphisme pour les automates de degré borné

On peut trouver dans [Boo78] une démonstration (non explicite) du fait que le problème d'isomorphisme pour les automates déterministes est polynomialement équivalent au problème d'isomorphisme[1] pour les graphes orientés finis. On prouve dans ce travail (Théorème 50) un résultat de même nature pour les automates non-déterministes en utilisant un principe d'encodage conservant des bornes sur le degrés de sortie. Ainsi, en combinant les résultats du théorème 50 et du lemme 49, il est possible de calculer, pour m fixé, la taille du groupe d'automorphisme d'un automate des classes $\mathsf{N}_m(n)$, $\mathsf{N}'_m(n)$ ou $\mathsf{N}_m(n)^\bullet$ et $\mathsf{N}'_m(n)^\bullet$ en un temps polynomial en n.

Théorème 50. *On considère m un nombre entier naturel fixé. Le problème d'isomorphisme pour un automate de N_m N'_m, $\mathsf{N}_m(n)^\bullet$ ou $\mathsf{N}'_m(n)^\bullet$ peut être résolu en un temps polynomial.*

Le principe de la preuve du théorème 50 s'appuie sur un encodage des automates non-déterministes ainsi que sur le résultat du théorème 43 [Luk82]. On développe ces arguments dans ce qui suit. On notera que la démonstration est constructive et pourrait donc fournir le moyen de construire un algorithme de calcul. Toutefois la grandeur des exposants mis en jeu interdit une implémentation efficace de cet algorithme. Cette remarque suggère de s'employer à un encodage plus fin qui permette d'optimiser les exposants. Dans ce travail on a toutefois préféré s'appuyer sur des techniques d'étiquetage qui ont fait la preuve de leur efficacité en pratique sur les graphes (cf décrites dans [Gal14] pour une étude récente). Ces techniques sont décrites dans la section suivante.

Développement de la preuve du théorème 50

On commence par associer à un automate non-déterministe un graphe. On considère h une bijection arbitraire de Σ sur $[\![1, k]\!]$ (h est simplement une fonction de comptage).

On note $G_{\mathcal{A}}$ le graphe fini (V, E) défini par :

- L'ensemble V des sommets du graphe $G_{\mathcal{A}}$ est défini par :

$$\begin{aligned} V \stackrel{\text{def}}{=} \; & Q \cup (I \cap F^c) \times \{1, \ldots, k+1\} \\ & \cup (F \cap I^c) \times \{1, \ldots, k+2\} \\ & \cup (I \cap F) \times \{1, \ldots, k+3\} \\ & \cup \{((p, a, q), i) \mid (p, a, q) \in \Delta \text{ et } 1 \leq i \leq h(a)\} \cup \Delta. \end{aligned}$$

- L'ensemble E des arcs du graphe $G_{\mathcal{A}}$ est défini par :

$$\begin{aligned} E \stackrel{\text{def}}{=} \; & \{(p, (p, a, q)) \mid (p, a, q) \in \Delta\} \\ & \cup \{((p, a, q)), q) \mid (p, a, q) \in \Delta\} \\ & \cup \{((p, a, q), ((p, a, q), 1)) \mid (p, a, q) \in \Delta\} \\ & \cup \{(((p, a, q), i), ((p, a, q), j)) \mid (p, a, q) \in \Delta \text{ et } 1 \leq i, j \leq h(a)\} \\ & \cup \{((q, i), (q, j)) \mid q \in I \cap F^c \text{ et } 1 \leq i, j \leq k+1\} \\ & \cup \{((q, i), (q, j)) \mid q \in F \cap I^c \text{ et } 1 \leq i, j \leq k+2\} \\ & \cup \{((q, i), (q, j)) \mid q \in F \cap I \text{ et } 1 \leq i, j \leq k+3\} \\ & \cup \{(q, (q, 1)) \mid q \in I \cup F)\}. \end{aligned}$$

1. Pour être précis il ne s'agit pas exactement du même problème d'isomorphisme, les automates peuvent dans ce résultat avoir des alphabets différents et les isomorphismes modifient aussi les étiquettes des transitions.

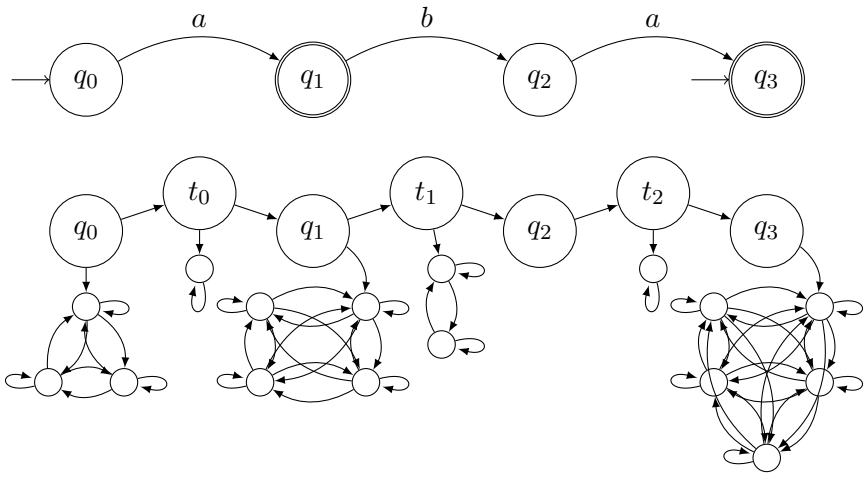


FIGURE 5.8 – Exemple de graphe associé à un automate, $h(a) = 1$ et $h(b) = 2$

Intuitivement, le principe consiste d'abord à décomposer chaque transition en deux arcs, un graphe complet (dont le nombre d'arcs dépend de la lettre de l'alphabet Σ mise en jeu dans la transition) est ensuite connecté au nœud intermédiaire.
Une construction du même type est est mise en œuvre pour les états initiaux (strictement), finals (strictement) ou à la fois initiaux et finals. La figure 5.8 illustre cette construction sur un exemple.

Suite Pour tout sommet s de $G_{\mathcal{A}} = (V, E)$, on note $d(s)$ le maximum de la taille d'une clique de $G_{\mathcal{A}}$ qui contient le sommet s.
Plus formellement, on a :

$$d(s) = \max\{|H|,\ H \times H \subseteq E \text{ and } s \in H\}|.$$

On peut noter que $d(s)$ est possiblement nul. Le résultat du lemme 51 donne un encadrement de $d(s)$.

Lemme 51. *Pour tout sommet s de $G_{\mathcal{A}}$, on a $0 \leqslant d(s) \leqslant k + 3$.*

Preuve. Par définition de $d(s)$. □

Lemme 52. *On considère $\mathcal{A}$ un automate fini. Pour tout arc de $G_{\mathcal{A}}$ de la forme (s, t), on a :*

1. *Si $d(s) = 0$, alors s est un élément de $Q \cup \Delta$,*
2. *Si $0 < d(s) \leqslant k$, alors s est de la forme $((p, \ell, q), i)$, avec $(p, \ell, q) \in \Delta$ et $1 \leq i \leq h^{-1}(\ell)$,*
3. *Si $d(s) = k + 1$, alors s est de la forme (q, i) avec $q \in I \cap F^c$,*
4. *Si $d(s) = k + 2$, alors s est de la forme (q, i) avec $q \in F \cap I^c$,*
5. *Si $d(s) = k + 3$, alors s est de la forme (q, i) avec $q \in F \cap I$.*

Preuve. Par définition de $d(s)$. □

Lemme 53. *On considère $\mathcal{A}$ Un automate fini. Pour tout arc de $G_{\mathcal{A}}$ de la forme (s, t) où $s \in Q$, on a :*

1. *$d(t) \in \{0, k + 1, k + 2, k + 3\}$,*
2. *Si $d(t) = 0$, alors $t \in \Delta$,*
3. *Si $d(t) = k + 1$, alors $s \in I \cap F^c$,*

4. *Si* $d(t) = k + 2$, *alors* $s \in F \cap I^c$,
5. *Si* $d(t) = k + 3$, *alors* $s \in F \cap I$.

On peut alors énoncé le résultat de la proposition 54.

Proposition 54. *Deux automates finis* $\mathcal{A}_1$ *et* $\mathcal{A}_2$ *sont isomorphes si et seulement si* $G_{\mathcal{A}_1}$ *et* $G_{\mathcal{A}_2}$ *sont isomorphes.*

Preuve. Par définition des graphes associés, si les automates $\mathcal{A}_1$ and $\mathcal{A}_2$ sont isomorphes alors les graphes $G_{\mathcal{A}_1}$ et $G_{\mathcal{A}_2}$ le sont aussi.

Réciproquement, on considère $\mathcal{A}_1 = (Q_1, A, \Delta_1, I_1, F_1)$ et $\mathcal{A}_2 = (Q_2, A, \Delta_2, I_2, F_2)$ deux automates non-déterministes tels que les graphes associés $G_{\mathcal{A}_1}$ et $G_{\mathcal{A}_2}$ soient isomorphes. On note φ un isomorphisme de $G_{\mathcal{A}_1}$ sur $G_{\mathcal{A}_2}$.
On peut remarquer que pour tout sommet s de $G_{\mathcal{A}_1}$, $d(\varphi(s)) = d(s)$. On a donc $d(s) = 0$ si et seulement si $d(\varphi(s)) = 0$. Le résultat du lemme 52 assure alors que φ induit une application bijective de Q_1 sur Q_2. La suite de la preuve consiste à démontrer que la restriction de φ à Q_1 définit une bijection de Q_1 sur Q_2.

- Si $q \in I_1 \cap F_1^c$, alors $d((q, 1)) = k + 1$. Par conséquent $d(\varphi((q, 1))) = k + 1$. Puisque $(q, (q, 1))$ est un arc de $G_{\mathcal{A}_1}$, $(\varphi(q), \varphi((q, 1)))$ est un arc de $G_{\mathcal{A}_2}$.
  L'assertion 3. du lemme 53 assure que $\varphi(q) \in I_2 \cap F_2^c$.
- Si $q \in I_1^c \cap F_1$, alors $d((q, 1)) = k + 2$. Par conséquent $d(\varphi((q, 1))) = k + 2$. Puisque $(q, (q, 1))$ est un arc de $G_{\mathcal{A}_1}$, $(\varphi(q), \varphi((q, 1)))$ est un arc de $G_{\mathcal{A}_2}$.
  L'assertion 4. du lemme 53 assure que $\varphi(q) \in I_2 \cap F_2^c$.
- Si $q \in I_1 \cap F_1$, alors $d((q, 1)) = k + 3$. Par conséquent $d(\varphi((q, 1))) = k + 3$. Puisque $(q, (q, 1))$ est un arc de $G_{\mathcal{A}_1}$, $(\varphi(q), \varphi((q, 1)))$ est un arc de $G_{\mathcal{A}_2}$.
  L'assertion 5. du lemme 53 assure que $\varphi(q) \in I_2 \cap F_2^c$.
- Si $(p, a, q) \in \Delta_1$, alors $d((p, a, q)) = 0$). Par conséquent $d(\varphi((p, a, q))) = 0$).
  Puisque $(p, (p, a, q))$ est un arc du graphe $G_{\mathcal{A}_1}$, $(\varphi(p), \varphi((p, a, q)))$ est un arc du graphe $G_{\mathcal{A}_2}$.
  L'assertion 2. du lemme 53 assure que $\varphi((p, a, q)) \in \Delta_2$.
  On pose dans la suite $\varphi((p, a, q)) = (s, b, t)$, avec $s, t \in Q_2$. Le seul arc arrivant au sommet (s, b, t) dans le graphe $G_{\mathcal{A}_2}$ est $(s, (s, b, t))$.
  Puisque φ est un isomorphisme, $(\varphi^{-1}(s), (p, a, q))$ est un arc de $G_{\mathcal{A}_1}$. On a donc $\varphi^{-1}(s) = p$.
  Il y a deux arcs sortant du sommet (s, b, t) du graphe $G_{\mathcal{A}_2}$: $((s, b, t), (1, (s, b, t))$ et $((s, b, t), t)$. Les deux arcs sortant de (p, a, q) dans $G_{\mathcal{A}_1}$ sont $((p, a, q), 1, (p, a, q))$ et $((p, a, q), q)$.
  Puisque $d(q) = 0$, $d(t) = 0$, $d((1, (p, a, q))) = h(a)$ et $d(((s, b, t), t)) = h(b)$, on a nécessairement $\varphi(q) = t$ et $h(a) = h(b)$ (et ainsi $a = b$).
  On a finalement démontrer que si $(p, a, q) \in \Delta_1$, alors $(\varphi(p), a, \varphi(q)) \in \Delta_2$.

La même démonstration peut être appliquée à φ^{-1}, démontrant ainsi la proposition. □

Proposition 55. *La taille de* $G_{\mathcal{A}}$ *est polynomiale en la taille de* $\mathcal{A}$. *De plus, si tout état de* $\mathcal{A}$ *admet au plus* m *transitions sortantes, le degré de* $G_{\mathcal{A}}$ *est majoré par* $\max\{m+1, k+3\}$.

Preuve. Par construction, la taille de $G_{\mathcal{A}}$ est polynomiale en la taille de $\mathcal{A}$. Soit s un sommet du graphe $G_{\mathcal{A}}$. On distingue les cas suivants :

- Si $s \in Q \cap I^c \cap F^c$, alors les arcs de $G_{\mathcal{A}}$ partant de s sont tous de la forme (s, t) où $t \in \Delta$, en tant que transition de $\mathcal{A}$, part de l'état s.
- Si $s \in I \cup F$, alors les arcs du graphe $G_{\mathcal{A}}$ partant de s sont ceux de la forme (s, t) où $t \in \Delta$, en tant que transition de $\mathcal{A}$, part de l'état s, ou l'arc de s à $(s, (s, 1))$.
  On peut donc conclure que, dans ce cas, il existe au plus $m + 1$ arcs sortant de s.
- Si $s \in \Delta$, il existe alors deux arcs sortant de s.

- Si $s = (q, i)$, avec $q \in Q$, alors il existe au plus $k + 3$ arcs sortant de (q, i).
- Si $s = (t, i)$, avec $t \in \Delta$, alors il existe au plus k arcs sortant de (t, i), où k dépend de la lettre de l'alphabet Σ mise en jeu dans la transition t.

Cette étude exhaustive des cas à considérer prouve le résultat de la proposition. □

Le résultat du théorème 50 est une conséquence des résultats des propositions 55, 54 et du théorème 43.

5.4.5 Calculs pratiques basés sur la technique d'étiquetage

Pour tester l'isomorphisme de graphes, l'approche la plus utilisée pour l'heure est basée sur des techniques d'étiquetage [Gal14]. Cette technique fonctionne de façon efficace pour des graphes de grande taille.
Il s'agit d'adapter ici cette façon de faire aux automates non-déterministes.
Chaque état d'un automate à n états appartient à une et une seule des quatre parties, déjà utilisées dans les sections précédentes, définies par l'appartenance ou non aux ensembles des états initiaux ou finals.

Le tableau 5.3 résume les situations et les notations employées.

Description	Ensembles	Notation de l'ensemble	Notation du cardinal
initial et final	$I \cap F$	$Q_1(\mathcal{A})$	$n_1(\mathcal{A})$
final mais pas initial	$F \cap I^c$	$Q_2(\mathcal{A})$	$n_2(\mathcal{A})$
initial mais pas final	$I \cap F^c$	$Q_3(\mathcal{A})$	$n_3(\mathcal{A})$
ni initial, ni final	$F^c \cap I^c$	$Q_4(\mathcal{A})$	$n_4(\mathcal{A})$

TABLE 5.3 – Appartenance d'un état d'un automate $\mathcal{A}$ à un sous-ensemble de l'ensemble Q de ses états

En utilisant les notations du tableau 5.3, on peut affirmer intuitivement que si deux automates $\mathcal{A}_1$ et $\mathcal{A}_2$ à n états sont isomorphes alors pour $i \in \{1, 2, 3, 4\}$, $n_i(\mathcal{A}_1) = n_i(\mathcal{A}_2)$ (on notera dans la suite n_i la valeur commune des deux cardinaux). Par ailleurs, si φ un isomorphisme de $\mathcal{A}_1$ sur $\mathcal{A}_2$, alors pour $i \in \{1, 2, 3, 4\}$, φ induit une bijection de $Q_i(\mathcal{A}_1)$ sur $Q_i(\mathcal{A}_2)$, de plus les quatre bijections induites définissent l'isomorphisme φ. Ainsi, plutôt que de tester les $n!$ permutations du groupe symétrique pour détecter les isomorphismes, suffit-il d'en tester $n_1! \cdot n_2! \cdot n_3! \cdot n_4!$.
Avec une répartition optimale des quatre ensembles qui partitionnent Q, le nombre de tests passe de $n!$ à $((n/4)!)^4$. On peut se donner une première idée du gain obtenu en évaluant un équivalent du rapport dans le cas d'une répartition optimale pour 4 étiquettes, grâce à la formule de Stirling :

$$\frac{n!}{((n/4)!)^4} \sim \sqrt{\frac{2^5}{\pi^3 n^3}} \cdot 4^n.$$

De façon générale, la répartition optimale pour p étiquettes, conduit à :

$$\frac{n!}{((n/p)!)^p} \sim \sqrt{\frac{p^p}{(2n\pi)^{p-1}}} \cdot p^n.$$

La courbe de la figure 5.9, montre que le gain réalisé par les techniques de *labelling*, est dans la répartition optimale pour 4 étiquettes, très décisif.
l'approche par étiquetage est illustrée par la figure 5.10.

Toute partition de l'ensemble des états des automates considérés, en un nombre plus grand de sous-ensembles, sur des critères faciles à mettre en œuvre sur le plan calculatoire

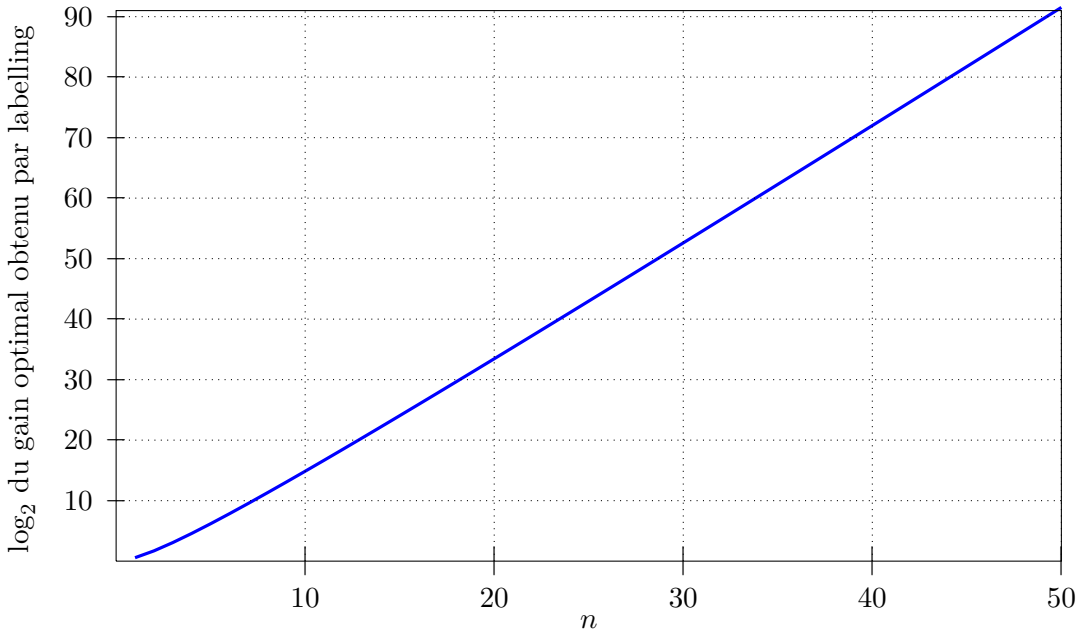


FIGURE 5.9 – Courbe représentative de $n \mapsto \log_2\left(n!/\left(\frac{n}{4}!\right)^4\right)$. Le gain obtenu par étiquetage est, dans le cas optimal, même pour de petites valeurs de n, décisif.

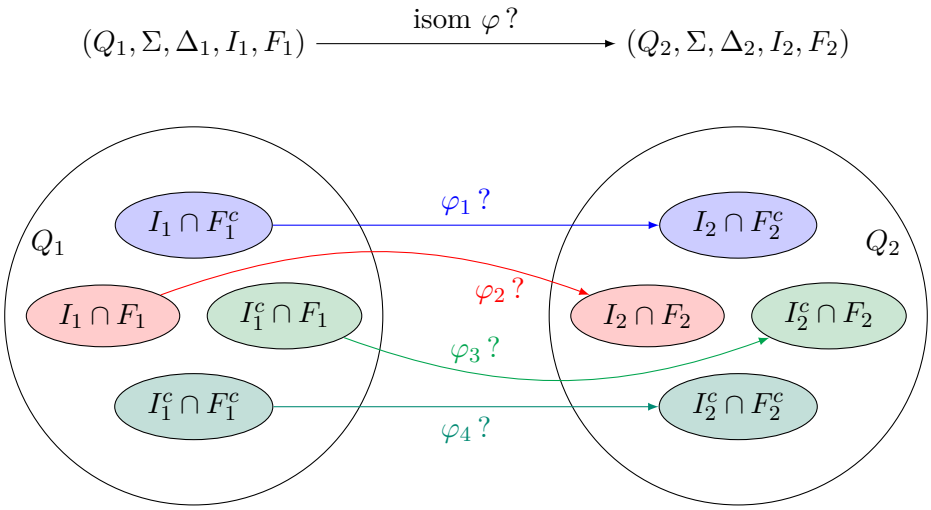


FIGURE 5.10 – Principe de la technique d'étiquetage

et surtout stables par isomorphisme, constitue une amélioration de l'idée qui vient d'être exposée. C'est exactement cette méthode qui est utilisée pour les graphes avec l'approche par étiquetage.

Cette méthode peut être directement adaptée aux automates finis.

Formalisons les idées précédentes : Un *étiquetage* est une application calculable τ de $\mathsf{N}(n) \times [\![1, n]\!]$ dans un ensemble fini D, telle que pour tous automates $\mathcal{A}_1 = (Q, \Sigma, \Delta_1, I_1, F_1)$ et $\mathcal{A}_2 = (Q, \Sigma, \Delta_2, I_2, F_2)$, si φ est un isomorphisme de $\mathcal{A}_1$ sur $\mathcal{A}_2$, alors :

$$\forall i \in [\![1, n]\!], \quad \tau(\mathcal{A}_1, i) = \tau(\mathcal{A}_2, \varphi(i))$$

L'idée est d'identifier (par une fonction de comptage par exemple) les éléments de Q et de $[\![1, n]\!]$.

Le principe de l'algorithme consiste alors à chercher des permutations φ qui respectent τ dans le sens donné par l'égalité précédente.
S'il existe $\alpha \in D$ tel que :

$$|\{i \mid \tau(\mathcal{A}_1, i) = \alpha\}| \neq |\{i \mid \tau(\mathcal{A}_2, i) = \alpha\}|.$$

alors les deux automates ne sont pas isomorphes. Si ce n'est pas le cas, toutes les permutations qui respectent τ sont testées.

Si dans le pire des cas, lorsque l'étiquetage ne dégrossit pas le test, il y a à nouveau $n!$ tests à réaliser, la méthode fonctionne néanmoins très bien en pratique. On peut également remarquer que si τ_1 et τ_2 sont deux étiquetages, alors τ défini par $\tau \stackrel{\text{def}}{=} (\tau_1, \tau_2)$ est également un étiquetage. On peut donc de combiner les différents étiquetages. Il existe d'autres techniques, comme du raffinement d'étiquetage, mais nous ne les avons pas mises en œuvre ici.

Étiquetages utilisés dans ce travail : Le tableau 5.4 présente de façon synthétique les étiquettes utilisées dans ce travail. Les automates considérés sont à n états, ils sont de la forme $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, où Q est identifié à $[\![1, n]\!]$.

Étiquettes	D	Description formelle de $\tau\,(\mathcal{A}, i)$
τ_1	$\mathbb{B}$	$i \in F$?
τ_2	$\mathbb{B}$	$i \in I$?
τ_3	$\mathbb{B}$	$i \in F \cap I$?
$\tau_{4,a}$	$[\![1,\mathrm{r}]\!]$	$\lvert\{j \in Q,\, (i,a,j) \in \Delta\}\rvert$
$\tau_{5,a}$	$[\![1,\mathrm{r}]\!]$	$\lvert\{j \in Q,\, (j,a,i) \in \Delta\}\rvert$
τ_6	$\Sigma^{\leqslant r}$	$\min\{\omega \in \Sigma^*,\, i(\omega) \in F\}$
τ_7	$\Sigma^{\leqslant r}$	$\min\{\omega \in \Sigma^*,\, init(\omega) = i\}$

TABLE 5.4 – Étiquettes adaptées et utilisées pour les automates

De façon moins formelle :

- $\tau_1\,(\mathcal{A}, i)$ teste si l'état i de l'automate $\mathcal{A}$ est initial ou non.
- $\tau_2\,(\mathcal{A}, i)$ teste si l'état i de l'automate $\mathcal{A}$ est final ou non.
- $\tau_3\,(\mathcal{A}, i)$ teste si l'état i de l'automate $\mathcal{A}$ est à la fois initial et final.
- $\tau_{4,a}\,(\mathcal{A}, i)$ renvoie, pour chaque lettre a de Σ le nombre de transitions, étiquetées par a, sortant de l'état i.
- $\tau_{5,a}\,(\mathcal{A}, i)$ renvoie, pour chaque lettre a de Σ le nombre de transitions, étiquetées par a arrivant à l'état i.
- $\tau_6\,(\mathcal{A}, i)$ renvoie le plus petit mot, pour l'ordre lexicographique, qui étiquette un chemin qui conduit de l'état i à un état final.
- $\tau_7\,(\mathcal{A}, i)$ renvoie le plus petit mot, pour l'ordre lexicographique, qui étiquette un chemin qui conduit d'un état initial à l'état i.

L'utilisation de cette technique conduit en pratique à un calcul relativement efficace de la taille des groupes d'automorphismes associés aux automates considérés.

5.4.6 Expériences

Les expériences ont été menées sur une machine personnelle équipée d'un processeur `IntelCore i3-4150 CPU 3.50GHz x 4, 7,7 Gio` de mémoire, dans un environnement 64 bits Ubuntu 14.04 OS. L'implémentation a été réalisée par un code non optimisé écrit en Python.

La première expérience consiste à mesurer le temps nécessaire pour se déplacer dans les chaînes de Markov associées aux deux classes $\mathsf{N}(n)$ and $\mathsf{N}_m(m)$.
Les résultats sont consignés dans les tableaux 5.5 et 5.6.
Les étiquettes utilisées pour ces calculs sont celles décrites dans le la section 5.4.5.

Pour l'échantillonnage, c'est le n^3-ième élément obtenu dans le déplacement qui est renvoyé à partir d'un point de départ arbitraire.
Ces premiers résultats montrent que le passage de 2 à 3 lettres dans l'alphabet utilisé ne semble pas avoir une influence significative sur le temps de calcul.
De même, le fait de majorer ou non le nombre de transitions sortantes par état ne semble pas non plus significatif.

Plusieurs points techniques méritent des améliorations. En particulier le code n'a pas été optimisé, plusieurs calculs sur les étiquettes pourraient être ré-utilisés dans les déplacements dans la chaîne. Une programmation plus dynamique est donc envisageable. On peut noter par ailleurs que Python (comparé à C ou Java) n'est sans doute pas le langage le mieux adapté à ce type de calculs. En pratique, le problème d'isomorphisme des graphes orientés, donne des résultats excellents pour des graphes de grande taille (cf [FPSV09]).
Il faut retenir aussi que le nombre de pas dans la chaîne (n^3) est un facteur clef de l'augmentation du temps de calcul (en fonction de n) : le temps moyen pour réaliser un seul pas est multiplié par facteur de 10 environ lorsque l'on passe de $n = 20$ à $n = 90$. Le calcul des tailles des groupes d'automorphismes est donc très rapide.

n	10	20	50	70	90
$\lvert\Sigma\rvert = 2$	0.02	0.43	32.5	166.1	569.9
$\lvert\Sigma\rvert = 3$	0.02	0.56	47.1	248.4	848.1

TABLE 5.5 – Temps moyen d'échantillonnage pour un automate de $\mathsf{N}(n)$

n	10	20	50	70	90
$m = 2, \lvert\Sigma\rvert = 2$	0.2	0.43	32.5	166.1	566.8
$m = 2, \lvert\Sigma\rvert = 3$	0.2	0.57	47.0	246.7	847.2
$m = 3, \lvert\Sigma\rvert = 2$	0.2	0.43	33.0	167.8	561.9
$m = 3, \lvert\Sigma\rvert = 3$	0.2	0.57	47.2	248.6	851.3

TABLE 5.6 – Temps moyen d'échantillonnage pour un automate de $\mathsf{N}'_m(n)$

Les dernières expériences proposent de comparer notre générateur pour les automates de $\mathsf{N}'_2(n)^\bullet$ avec le générateur proposé dans [TV05] avec une densité de a-transitions de 2 et 3. Les paramètres de l'agorithme mis en concurrence sont la probabilité p_f qu'un état soit final et la densité σ des a-transitions. Plus clairement, l'ensemble des états de l'automate est $[\![1, n]\!]$, l'état 1 est l'unique état initial, on décide pour chaque état de son caractère final avec un tirage de Bernoulli de probabilité p_f. Enfin, pour tout état p et toute lettre a on décide si (p, a, q) est une transition avec la probabilité $\frac{\sigma}{n}$. Ainsi pour chaque lettre et chaque état l'espérance de la variable aléatoire qui compte le nombre de transitions sortantes est-elle, pour la lettre et l'état considérés, égale à σ.

L'algorithme a été lancé avec les paramètres $p_f = 0.2$ et $\sigma \in \{2, 3\}$. Pour chaque taille on calcul la taille moyenne de s de l'automate minimal associé. L'expérience a été menée avec des automates sur un alphabet à 2 lettres, la taille moyenne a été calculée en engendrant 1000 automates dans chaque cas. Les tableaux 5.7 et 5.8 donnent les résultats obtenus.

$\sigma = 2$						
n	5	8	11	14	17	20
s	1.3	3.0	4.8	5.1	4.5	4.0
$\sigma = 3$						
n	5	8	11	14	17	20
s	2.8	4.8	4.7	3.8	3.4	3.0

TABLE 5.7 – Taille moyenne de l'automate minimal déterministe associé aux automates engendrés par le générateur [TV05]

.

$\mathsf{N}'_2(n)^\bullet$						
n	5	8	11	14	17	20
s	3.7	6.1	7.9	10.0	11.5	13.9

TABLE 5.8 – Taille moyenne de l'automate minimal déterministe associé aux automates engendrés dans $\mathsf{N}'_2(n)^\bullet$

.

On peut observer que les deux générateurs fournissent des automates aux caractéristiques statistiques différentes.
L'approche par chaîne de Markov conduit semble-t-il à des automates pour lesquels la taille de l'automate minimal associé est plus grande.

Pour comparer les générateurs, nous avons, enfin, lancé une dernière expérience pour comparer la tailles des groupes d'automorphismes obtenus. Rappelons qu'avec la méthode par chaînes de Markov, la taille des groupes d'automorphismes est très proche de 1 en moyenne (voir Table 5.2, obtenues sur la chaîne sans la biaiser par l'algorithme de Metropolis Hastings, mais les résultats sont similaires pour la génération à isomorphisme près). Le tableau 5.9 rapporte la taille moyennes des groupes d'automorphismes obtenus à l'aide de la méthode de [TV05]. Les différences sont significatives.

5.5 Conclusions et perspectives

Ce chapitre fait suite aux travaux antérieurs [CF12, DFN13] portant sur la génération aléatoire de certaines classes d'automates en s'appuyant sur une méthode de Monte-Carlo par Chaîne de Markov.

Au delà de la contribution qui consiste à construire un générateur pour des automates non-déterministes, on montre surtout comment une application de l'algorithme de Metropolis-Hastings permet d'obtenir un générateur uniforme *à isomorphisme près*.

Les sections 5.4.3, 5.4.4 et 5.4.5 sont techniques mais dignes d'intérêt. Elles proposent une étude des classes d'équivalence des automates non-déterministes à partir des groupes

	$p_f, p_i = 0.2$ $\sigma = 2$	$p_f, p_i = 0.5$ $\sigma = 2$	$p_f, p_i = 0.8$ $\sigma = 2$	$p_f, p_i = 0.2$ $\sigma = 3$	$p_f, p_i = 0.5$ $\sigma = 3$	$p_f, p_i = 0.8$ $\sigma = 3$
$n = 5$	26	6	25	21	6	24
$n = 8$	2345	112	2213	2254	102	2441
$n = 10$	86 500	1343	71472	83072	1303	79203

TABLE 5.9 – Taille moyenne des tailles des groupes d'automorphismes par la méthode [TV05], avec un alphabet à 2 lettres.

.

d'isomorphismes associés à chacun de ces automates. On montre comment des techniques d'étiquetage permettent un calcul efficace des tailles des groupes d'automorphismes.

Un des buts de cet exposé est de convaincre de la souplesse d'utilisation des techniques de type MCMC pour échantillonner des classes pour lesquelles le caractère spécifiable (au sens de la méthode symbolique) ne représente pas un enjeu. Nous espérons que ce but a été un tant soit peu atteint.

Il est clair que le talon d'Achille de la méthode réside dans la difficulté de l'évaluation du temps de mélange. Le résultat de la section 5.3.2 qui donne une majoration pour la version paresseuse de la chaîne dans le cas de la classe la plus générale réalise un premier pas dans le bon sens. Il est par ailleurs certainement possible de ré-investir la technique qui consiste à *plonger* la chaîne étudiée dans un produit d'hypercubes, dans d'autres situations.

De façon plus générale, on peut sans doute envisager, pour des générateurs MCMC d'automates à n états, des marches aléatoires dans des graphes de diamètres en $\mathcal{O}(n)$ pour lesquels le temps de mélange ne peut guère être espéré inférieur à $\Theta(n^2)$. Le résultat conjecturé en $\mathcal{O}\left(n^3\right)$ semble donc constituer déjà un résultat relativement optimiste. Il interdit cependant d'envisager raisonnablement le tirage aléatoire d'automates de plus de quelques milliers d'états (car il faut prendre aussi en compte le temps de traitement de chaque étape).

Les prolongements naturels à ce chapitre pourraient consister à :

- Définir plus finement des classes *intéressantes* d'automates en commençant par donner un sens un peu formel à l'adjectif.
- Explorer plus finement les techniques de *couplage depuis le passé* et de *strong stationnary times*, soit pour majorer les temps de mélange dans les cas particulier, soit pour obtenir des générateurs parfaits (sans garantie de temps de terminaison) mais sans avoir à calculer des temps de mélange.
- Envisager des expériences portant sur les classes d'automates engendrés afin d'utiliser les générateurs construits.

Chapitre 6

Génération aléatoire d'automates déterministes, accessibles et partiellement ordonnés : estimation statistique des temps de mélange

Sommaire

6.1 Introduction

6.1.1 Cadre et intérêt

Rappelons (voir section 1.2) qu'un automate partiellement ordonné est un automate dont les circuits ne peuvent être que des boucles. Ces automates interviennent, dans le cadre non déterministe, dans plusieurs contextes : le *regular model-checking* [BMT01, CHM08], la classification des langages régulier [Arf87, STV01], la complexité des représentations par expressions régulières [Hó3], etc.

Sur le plan de la génération aléatoire, les travaux présentés ici sont très proches de ceux de [CF12] sur les automates acycliques. Nous souhaitons ici à la fois nous concentrer sur une classe un peu plus large, mais aussi montrer la souplesse d'utilisation des chaînes de Markov pour la génération aléatoire. L'objectif est surtout de montrer la mise en œuvre de techniques statistiques pour valider (mais non prouver) des temps d'arrêt dans la marche aléatoire.

Construire une chaîne de Markov sur un univers E, en définissant les règles de transitions entre les différents éléments de E, dans le but de produire un générateur aléatoire uniforme, constitue une étape relativement facile à mettre en œuvre dans de nombreux cas.
La pierre d'achoppement de cette approche est souvent théorique : on butte sur l'estimation du temps de mélange de la chaîne. Les méthodes à base de chaînes de Markov, parce qu'elles sont relativement faciles à mettre en œuvre et implémentées dans de nombreux outils, sont néanmoins très utilisées, notamment pour la simulation numérique. Afin de contourner le problème du calcul du temps de mélange, des tests statistiques peuvent être mis en place afin de se convaincre (mais non de prouver) que l'on s'est suffisamment déplacé dans la chaîne, on pourra consulter [CC96] pour une étude générale sur ces questions. Dans ce travail, on présente une approche statistique destinée à montrer comment il est possible d'apprécier expérimentalement l'efficacité de la chaîne. Cette méthode s'appuie sur l'idée du test du χ^2 de Pearson. Il s'agit donc de définir une partition de l'ensemble E des éléments que l'on souhaite produire par l'échantillonneur pour laquelle la proportion de chacun des ensembles qui définissent la partition est connue ou accessible.

Notre étude porte sur les automates déterministes accessibles et partiellement ordonnés. Nous montrons tout d'abord comment les générer par chaînes de Markov, en étendant (facilement) les résultats de [CF12] ; nous étudions aussi la sous-classe des automates partiellement ordonnés avec au plus m boucles, le cas $m = 0$ correspondant au cas acyclique étudié dans [CF12] . Dans le cadre des automates partiellement ordonnés, une idée consiste à construire une partition en regroupant les automates qui comptent le même nombre de boucles. La combinatoire des automates accessibles, déterministes, partiellement ordonnés à n états et k boucles peut effectivement être déterminée par des techniques classiques qui sont détaillées dans la section 6.1.2 suivante.

Dans la suite de ce chapitre, sauf mention explicite contraire, les automates considérés sont déterministes.

6.1.2 Calcul du nombre de boucles

Dans la suite, $\mathbb{P}_n$ désigne l'ensemble des automates déterministes, accessibles, partiellement ordonnés sans états finals sur un alphabet fixé Σ. L'étude qui suit est une adaptation des travaux de [DFN13] à propos de la combinatoire des automates déterministes acycliques. Ces travaux s'appuient sur la notion de *sources secondaires*.

Sources secondaires

L'idée intuitive de *source* consiste simplement à reconnaître l'ordre dans lequel un état est atteint. Plus formellement, on considère $x \stackrel{\text{def}}{=} (\llbracket 1, n \rrbracket, \Sigma, \delta, \{1\})$ un élément de $\mathbb{P}_n$ pour lequel l'ensemble des états Q est l'intervalle de nombres entiers $\llbracket 1, n \rrbracket$ et 1 est l'état initial.

- On dit que l'état 1 est la seule 1-source ou la source primaire.
- On dit que $p \in \llbracket 1, n \rrbracket$ est une *source secondaire* ou une 2-source si $p \neq 1$ et si toutes les transitions qui arrivent en p et qui ne sont pas des boucles, ont pour origine l'état initial 1. Plus formellement, p est une 2-source si et seulement si les conditions suivantes sont réalisées :

  (1) $p \neq 1$.

  (2) $\forall (q, a) \in \llbracket 1, n \rrbracket \times \Sigma, \quad ((q, a, p) \in \delta \ \wedge \ q \neq p) \Rightarrow (q = 1)$
- On considère s un nombre entier naturel non-nul et on suppose la notion de k-source définie pour tout $k \in \llbracket 1, s \rrbracket$. On notera $\mathcal{S}_k$ l'ensemble des k-sources de l'automate x. On dit que $p \in \llbracket 1, n \rrbracket$ est une $(s+1)$-source si et seulement si les conditions suivantes sont réalisées :

  (1) $\forall k \in \llbracket 1, s \rrbracket, \quad p \notin \mathcal{S}_k$.

(2) $\forall (q,a) \in [\![1,n]\!] \times \Sigma, \quad \Big((q,a,p) \in \delta \ \wedge \ q \neq p \ \wedge \ q \notin \bigcup_{k \in [\![1,s-1]\!]} \mathcal{S}_k \Big) \Rightarrow (q = \mathcal{S}_s)$

La figure 6.1 illustre sur un exemple cette idée.

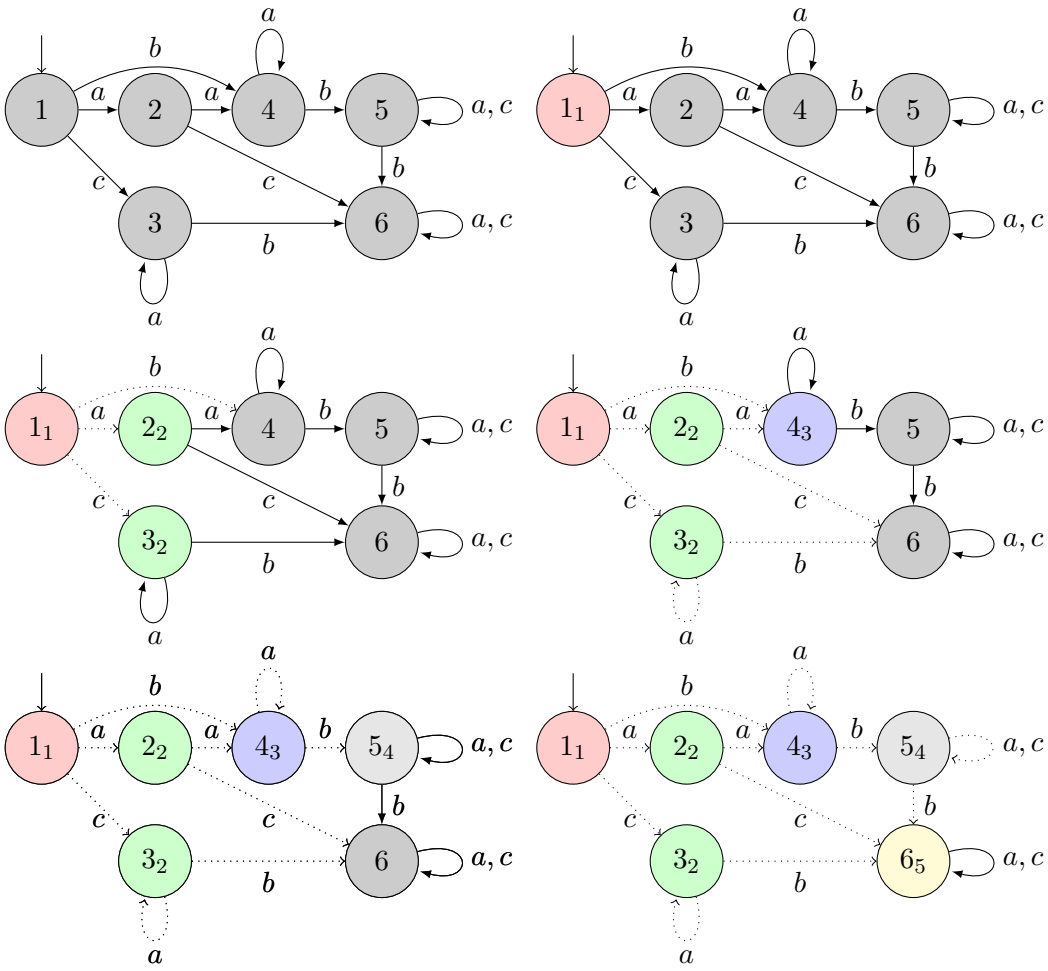


FIGURE 6.1 – Un automate partiellement ordonné à 6 états et le calcul de ses sources.

Proposition 56 (sources)**.** *Pour tout* $x \stackrel{def}{=} ([\![1,n]\!], \Sigma, \delta, \{1\}) \in \mathbb{P}_n$*, pour tout* $p \in [\![1,n]\!]$*, il existe un nombre entier strictement positif* k *tel que* p *soit une* k*-source.*

Preuve. On considère s un nombre entier strictement positif tel que $\bigcup_{k=1}^{s} \mathcal{S}_k$ soit strictement inclus dans $[\![1,n]\!]$. Pour tout état $r \in [\![1,n]\!] \setminus \bigcup_{k=1}^{s} \mathcal{S}_k$ l'ensemble des transitions de la forme (p,a,r) où $(p,a) \in [\![1,n]\!] \times \Sigma$ n'est pas vide, sinon x ne serait pas accessible. Parmi les transitions de la forme (p,a,r) qui ne sont pas des boucles, il existe au moins une transition telle que $p \notin \bigcup_{k=1}^{s-1} \mathcal{S}_k$ sinon r appartiendrait à $\bigcup_{k=1}^{s} \mathcal{S}_k$. Si pour tout état $r \in [\![1,n]\!] \setminus \bigcup_{k=1}^{s} \mathcal{S}_k$ il existait une transition (p,a,r) qui ne soit pas une boucle et telle que $p \in [\![1,n]\!] \setminus \bigcup_{k=1}^{s} \mathcal{S}_k$, puisque $[\![1,n]\!] \setminus \bigcup_{k=1}^{s} \mathcal{S}_k$ est fini, on pourrait obtenir un cycle de longueur strictement supérieur à 1 dans x, ce qui contredirait son caractère partiellement ordonné. Il existe donc $r \in [\![1,n]\!] \setminus \bigcup_{k=1}^{s} \mathcal{S}_k$ telles que les seules transitions de la forme (p,a,r) qui ne soit pas des

boucles vérifient $p \in \mathcal{S}_s$. L'ensemble $\mathcal{S}_{s+1}$ est donc non-vide. Puisque l'ensemble des états de x est fini, le plus petit nombre entier s tel que $\mathcal{S}_{s+1} = \emptyset$ vérifie donc : $\bigcup_{k=1}^{s} \mathcal{S}_k = [\![1, n]\!]$. □

La notion de sources permet d'associer à x un arbre $\sigma(x)$, appelé *arbre des sources*. Pour tout état q, il existe un nombre entier naturel non-nul k tel que q soit une k-source. On notera q_k à la place de q pour indiquer que q est une k-source. Si $k > 1$, l'ensemble des transitions de la forme (p_{k-1}, a, q_k) est non vide. En munissant l'ensemble des états et l'alphabet Σ d'une relation d'ordre, on peut émonder x en ne conservant, pour chaque état q_k, que la plus petite des transitions de la forme (p_{k-1}, a, q_k). Après émondage, pour tout nombre entier $k > 1$, tout état q qui est une k-source admet une unique transition entrante.

Proposition 57. *Pour tout x, $\sigma(x)$ est un arbre enraciné en 1.*

Preuve. Par construction de $\sigma(x)$ et en s'appuyant sur le résultat de la proposition 56. □

On retrouvera la notion d'arbre des sources dans la démonstration du lemme 62.

La figure 6.2 représente un automate partiellement ordonné et l'arbre des sources associé.

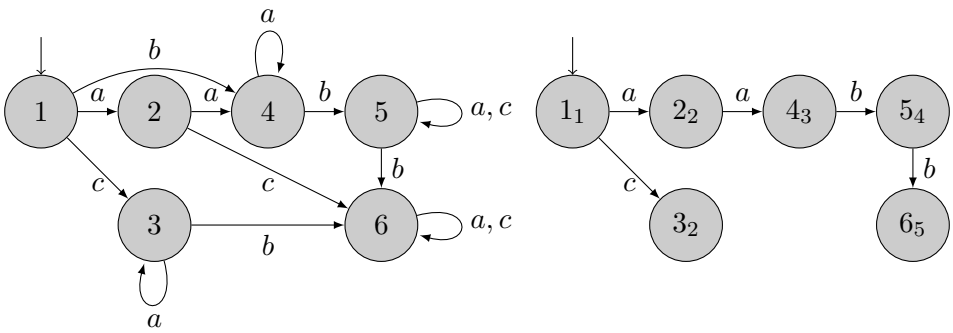


FIGURE 6.2 – Un automate partiellement ordonné à 6 états et l'arbre des sources associé.

En notant $b_k(n)$ le nombre d'automates à n états, k lettres, partiellement ordonnés, on a :

$$b_k(n) = \sum_{t=0}^{n-1} \binom{n}{t} (-1)^{n-t-1} (t+2)^{k(n-t)} b_k(t) \tag{6.1}$$

En notant maintenant $\eta_k(n$ le nombre d'automates partiellement ordonnés ayant 1 comme état initial, n états, k lettres, on a :

$$\eta_k(n) = \sum_{i=0}^{n-1} \binom{n-1}{i} (-1)^i b_k(n-1-i)(n-i+1)^{k(1+i)} \tag{6.2}$$

Ces formules sont des adaptations faciles des celles établies dans [DFN13] pour les automates acycliques.

Dans les équations **??** et 6.2, il suffit de marquer les boucles par la lettre u pour obtenir une expression polynomiale en u :

$$b_k(n,u) = \sum_{t=0}^{n-1} \binom{n}{t} (-1)^{n-t-1} (t+1+u)^{k(n-t)} b_k(t,u) \tag{6.3}$$

$$\eta_k(n,u) = \sum_{i=0}^{n-1} \binom{n-1}{i} (-1)^i b_k(n-1-i,u)(n-i+u)^{k(1+i)} \tag{6.4}$$

La table 6.1 montrent les résultats pour des automates accessibles, déterministes à n-états ($n \in [\![2, 5]\!]$) et 2 lettres.

	Nombre de Boucles							
Nombre d'états	0	1	2	3	4	5	6	Total
2	3	8	7	2	0	0	0	20
3	32	100	114	56	10	0	0	312
4	762	2640	3528	2268	702	84	0	9984
5	32712	122064	181848	138240	56640	11904	1008	544416

TABLE 6.1 – Nombres d'automates déterministes partiellement ordonnés, en fonction du nombre de boucles, pour un nombre d'états fixé de 2 à 5, pour un alphabet à 2 lettres

6.2 Chaîne de Markov, générateur

6.2.1 Génération aléatoire par chaîne de Markov

Il s'agit ici de construire un générateur aléatoire qui s'appuie sur la méthode de Monte-Carlo par Chaîne de Markov exposée dans la section 3.1. On reprend ici la terminologie et les notations de cette section de présentation. Ici l,'espace des états Ω est, pour n nombre entier naturel fixé donnant le nombre d'états de l'automate, $\mathbb{P}_n$. On décide, pour chaque état, par tirage de Bernoulli, s'il est final ou non.

Principes de déplacement d'un automate de $\mathbb{P}_n$ à un autre

Les déplacements dans la chaîne à partir d'un sommet x de Ω vont s'appuyer sur les principes qui suivent.

On considère $x \in \mathbb{P}_n$, les transitions dans la chaîne sont définies par le tirage aléatoire d'un triplet (p, a, q) de $Q \times \Sigma \times Q$, en examinant les cas suivants :

- Le triplet (p, a, q) est une transition de x. On considère alors y l'automate obtenu à partir de x en supprimant la transition (p, a, q). Si $y \in \mathbb{P}_n$ alors $x * (p, a, q)$ est défini par $x * (p, a, q) \stackrel{\text{def}}{=} y$. Si $y \notin \mathbb{P}_n$ (typiquement y n'est pas accessible) alors $x * (p, a, q)$ est défini par $x * (p, a, q) \stackrel{\text{def}}{=} x$.
- Le triplet (p, a, q) n'est pas une transition de x et par ailleurs, il n'existe aucune transition de la forme (p, a, r) dans l'automate x où $r \in [\![1, n]\!]$. On considère alors l'automate y obtenu à partir de x en ajoutant la transition (p, a, q). Comme précédemment, si $y \in \mathbb{P}_n$ alors $x * (p, a, q)$ est défini par $x * (p, a, q) \stackrel{\text{def}}{=} y$. Si $y \notin \mathbb{P}_n$ (typiquement y n'est plus partiellement ordonné) alors $x * (p, a, q)$ est défini par $x * (p, a, q) \stackrel{\text{def}}{=} x$.
- Le triplet (p, a, q) n'est pas une transition de x et il existe par ailleurs, dans l'automate x, une transition de la forme (p, a, r), avec $r \in [\![1, n]\!] \backslash \{q\}$. On considère alors l'automate y obtenu à partir de x en supprimant la transition (p, a, r) et en ajoutant la transition (p, a, q). Comme précédemment, si $y \in \mathbb{P}_n$ alors $x * (p, a, q)$ est défini par $x * (p, a, q) \stackrel{\text{def}}{=} y$. Si $y \notin \mathbb{P}_n$ alors $x * (p, a, q)$ est défini par $x * (p, a, q) \stackrel{\text{def}}{=} x$.

La figure 6.3 illustre ces trois situations.

Matrice de transition

La construction de la matrice de transition M sur l'ensemble $\mathbb{P}_n$, s'appuie sur la définition précédente de l'opérateur $*$.

Pour tout couple d'automates **distincts** (x, y) de $\mathbb{P}_n$, s'il existe un triplet (p, a, q) tel

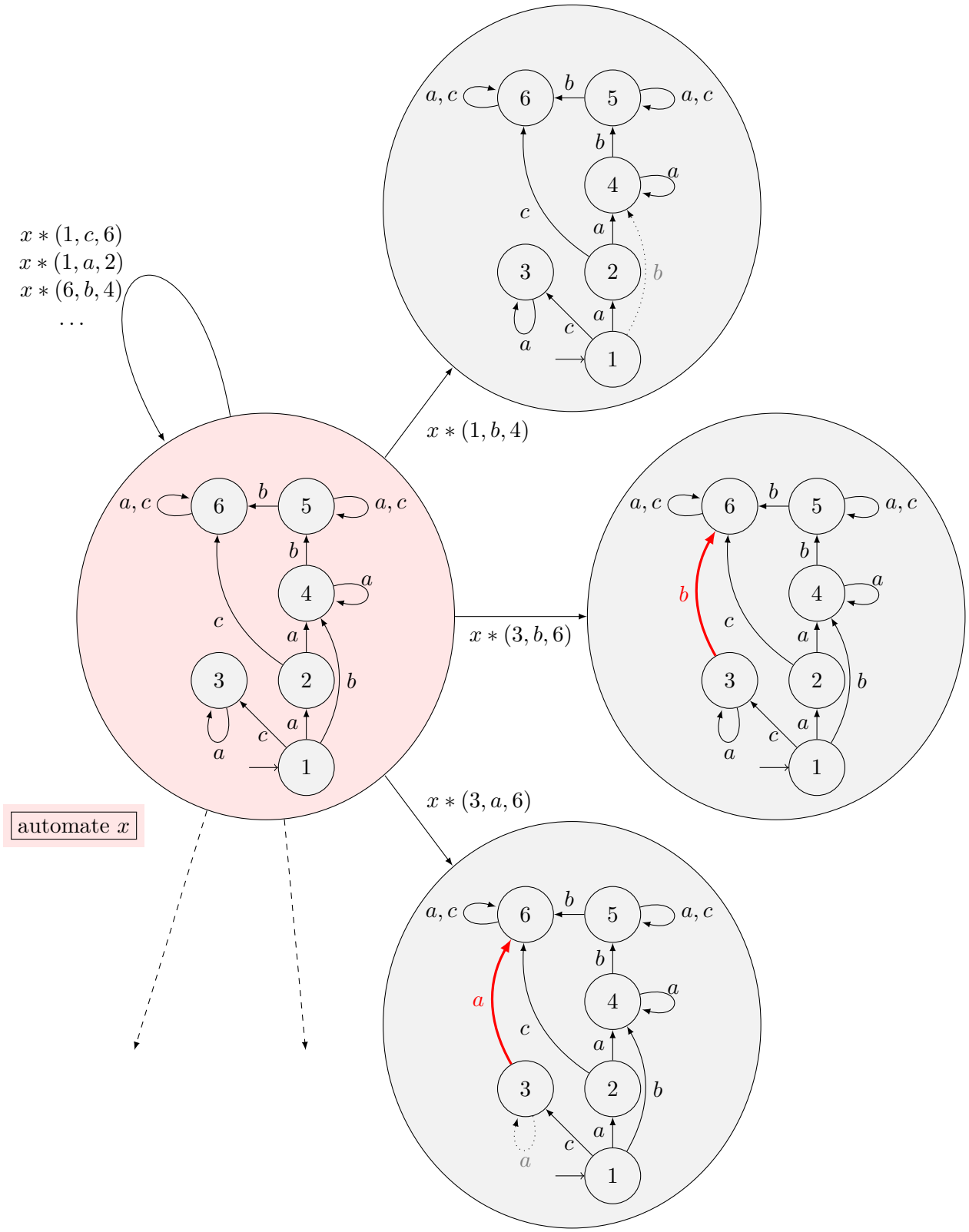


FIGURE 6.3 – Mouvement dans la chaîne $\Omega = \mathbb{P}_n$. Les flèches pointillées symbolisent les autres mouvements possibles dans la chaîne. On a représenté quelques triplets qui conduisent à une boucle sur l'automate x. $x * (1, c, 6)$ et $x * (1, a, 2)$ conduiraient à un automate qui ne serait plus accessible, $x * (6, b, 4)$ conduirait à un automate qui ne serait plus partiellement ordonné.

que $y = x * (p, a, q)$, on pose :

$$M(x, y) \stackrel{\text{def}}{=} \frac{1}{|\Sigma|n^2}$$

Dans le cas contraire, toujours avec $x \neq y$, si pour tout triplet (p, a, q), $y \neq x * (p, a, q)$, on pose $M(x, y) \stackrel{\text{def}}{=} 0$.

Afin que la matrice M soit stochastique, la définition de $M(x, x)$ est contrainte :

$$M(x, x) \stackrel{\text{def}}{=} 1 - \sum_{y \in \mathbb{P}_n \setminus \{x\}} M(x, y)$$

Puisque l'on peut considérer $|\Sigma|n^2$ triplets il existe au plus $|\Sigma|n^2$ automates y différents de x tels que $M(x, y) \neq 0$. Puisque tous les coefficients non-nuls et non diagonaux de la matrice M ont la même valeur $\frac{1}{|\Sigma|n^2}$, le coefficient diagonal $M(x, x)$ est donc positif.

On peut également remarquer que si l'automate x a au moins deux états, il existe une transition de la forme (p, a, q) où $p \neq q$. En effet si toutes les transitions sont des boucles l'automate x ne peut pas être accessible.

Puisque x est sans cycle, le triplet (q, a, p) n'est pas une transition de x et l'automate $x * (q, a, p)$ est alors défini par $x * (p, a, q) \stackrel{\text{def}}{=} x$. En particulier :

$$\sum_{y \in \mathbb{P}_n \setminus \{x\}} M(x, y) < 1$$

et $M(x, x)$ est donc strictement positif.

On peut reformuler les choses en disant que puisqu'il existe au moins un triplet tel que $x * t = x$, il existe donc une boucle à chaque sommet du graphe associé à la chaîne de Markov construite. Tous les états de l'espace Ω sont donc de période 1.

Les résultats qui suivent poursuive l'étude de la chaîne de Markov construite et sont destinés à démontrer en particulier son ergodicité.

Vers l'ergodicité

Les deux premiers lemmes qui suivent prouvent que M est symétrique.

Lemme 58. *On considère $x \in \mathbb{P}_n$ et (p, a, q) un triplet de $Q \times \Sigma \times Q$.*
Si $x * (p, a, q) \neq x$ alors il existe $q' \in Q$ tel que $(x * (p, a, q)) * (p, a, q') = x$.

Preuve. Ce résultat apparaît clairement comme une conséquence directe de l'opérateur $*$. En effet si $x * (p, a, q) \neq x$, on est dans la situation où l'opération correspond à une suppression, un ajout ou une redirection. On précise les trois cas :

(1) La transition (p, a, q) existe dans l'automate x et sa suppression est possible. L'opération $(x * (p, a, q)) * (p, a, q)$ consiste donc simplement à ajouter la transition (p, a, q) à l'automate $x * (p, a, q)$ obtenu par suppression de la transition (p, a, q). On revient donc à x.

(2) La transition (p, a, q) n'existe pas dans l'automate x et aucune transition de la forme $(p, a, ?)$ n'existe dans x. L'automate $(x * (p, a, q))$ est donc obtenu par ajout de la transition (p, a, q). L'opération $(x * (p, a, q)) * (p, a, q)$ consiste donc à supprimer la transition (p, a, q). On revient donc à x.

(3) La transition (p, a, q) n'existe pas dans l'automate x mais il existe une transition de la forme (p, a, r) cette transition est donc remplacée par la transition (p, a, q), l'opération $(x * (p, a, q)) * (p, a, r)$ consistera alors à remplacer la transition (p, a, q) de l'automate $x * (p, a, q)$ par la transition (p, a, r), on revient donc à x.

Dans tous les cas, si $x*(p,a,q) \neq x$, il existe $q' \in Q$ tel que $(x*(p,a,q))*(p,a,q') = x$. □

Lemme 59. *Pour tout couple d'automates $(x,y) \in \mathbb{P}_n^2$, on a :*

$$M(x,y) = M(y,x)$$

la chaîne de Markov associée est donc symétrique.

Preuve. On a remarqué que pour $n \geqslant 2$, les coefficients diagonaux de la matrice M sont strictement positifs. La chaîne est donc apériodique. Par ailleurs le résultat du lemme 58 assure que pour tout $x, y \in \mathbb{P}_n$ tels que $x \neq y$, si $M(x,y) \neq 0$ alors $M(y,x) \neq 0$. Puisque tous les coefficients non-diagonaux et non-nuls de la matrice M sont égaux à $\frac{1}{|\sigma|n^2}$, la matrice M est donc symétrique. □

Nous avons progressé vers l'ergodicité. Un pas de plus sera franchi lorsque l'irréductibilité de la chaîne aura été démontrée. Ce résultat nécessite la construction de quelques outils.

Définition 60. *On considère $x = (\llbracket 1, n \rrbracket, \Sigma, \delta, \{1\}) \in \mathbb{P}_n$, on dit que x est sérié s'il existe au plus une transition sortante pour chaque état et si δ ne contient aucune boucle.*

On peut facilement démontrer par récurrence sur le nombre n d'états des automates considérés que si x est sérié alors l'ordre sous-jacent est total et on peut représenter x par :

$$1 = p_0 \overset{a_1}{\to} p_1 \overset{a_2}{\to} \dots \overset{a_{n-1}}{\to} p_{n-1}$$

où les a_i sont des éléments de l'alphabet Σ et les $\{p_i,\ i \in \llbracket 0, n-1 \rrbracket\} = \llbracket 1, n \rrbracket$.

Le caractère irréductible de la chaîne est directement lié aux deux résultats qui suivent :
- Il existe un chemin entre deux automates sériés quelconques de $\mathbb{P}_n$.
- Il existe un chemin entre un automate quelconque de $\mathbb{P}_n$ et un automate sérié de $\mathbb{P}_n$.

Dans le lemme qui suit G désigne le graphe associé à la chaîne de Markov.

Lemme 61 (chemin entre automates sériés)**.** *On considère $x, y \in \mathbb{P}_n$ deux automates sériés. Il existe un chemin de x à y dans G.*

Preuve. On considère x et y, deux automates sériés distincts de $\mathbb{P}_n$. On suppose que x et y sont respectivement représentés par :

$$x : 1 = p_0 \overset{a_1}{\to} p_1 \overset{a_2}{\to} \dots \overset{a_{n-1}}{\to} p_{n-1}, \quad y : 1 = q_0 \overset{b_1}{\to} q_1 \overset{b_2}{\to} \dots \overset{b_{n-1}}{\to} q_{n-1}$$

On note alors $i = \daleth(x,y)$ [1] le plus petit indice tel que $(a_i, p_i) \neq (b_i, q_i)$.
Avant i, les deux automates coïncident donc, en particulier pour tout indice k strictement plus petit que i, on a $p_k = q_k$.

Les deux cas suivants peuvent alors se produire :
- Il existe $r \in Q$ tel que $p_i = q_i = r$. Alors nécessairement $a_i \neq b_i$ l'automate z défini par $z \overset{\text{def}}{=} x * (p_{i-1}, b_i, r) * (p_{i-1}, a_i, r)$ est alors sérié et vérifie $\daleth(z,y) > \daleth(x,y)$.
- $p_i \neq q_i$, on peut donc considérer $j \in \llbracket 1, n \rrbracket \backslash \{i\}$ tel que $q_i = p_j$. En fait, compte-tenu de la remarque faite précédemment à propos des indices strictement plus petits que i, on a $j > i$.
  Il faut alors considérer deux sous-cas possibles :
  * Si $a_i \neq b_i$, pour toute lettre $a \in \Sigma$, l'automate z défini par :

$$z \overset{\text{def}}{=} x * (p_{i-1}, b_i, p_j) * (p_{j-1}, a_j, p_j) * (p_{n-1}, a, p_i) * (p_{i-1}, a_i, p_i)$$

  est sérié et vérifie $\daleth(z,y) > \daleth(x,y)$.

1. daleth de (x,y)

∗ Si $a_i = b_i$, on peut considérer une lettre c de Σ, distincte de a_i ; l'automate x' défini alors par :

$$x' \stackrel{\text{def}}{=} x * (p_{i-1}, c, p_i) * (p_{i-1}, a_i, p_i)$$

permet de se ramener au premier cas.

Le processus décrit permet donc de construire une suite (x_i) d'automates sérialisés telle que $x_0 = x$ et telle que si $x_i \neq y$, il est possible de construire un chemin dans G de x_i à x_{i+1} où x_{i+1} est un automate sérié vérifiant :

$$\urcorner(x_{i+1}, y) > \urcorner(x_i, y)$$

Puisque pour tout couple (x, y) d'éléments distincts de $\mathbb{P}_n$, on a $\delta(x, y) \leqslant n - 1$, la suite (x_i) stationne en moins de $n - 1$ pas sur y. Ce qui prouve le résultat souhaité. □

La figure 6.4 donne un exemple de passage d'un automate sérié à un autre.

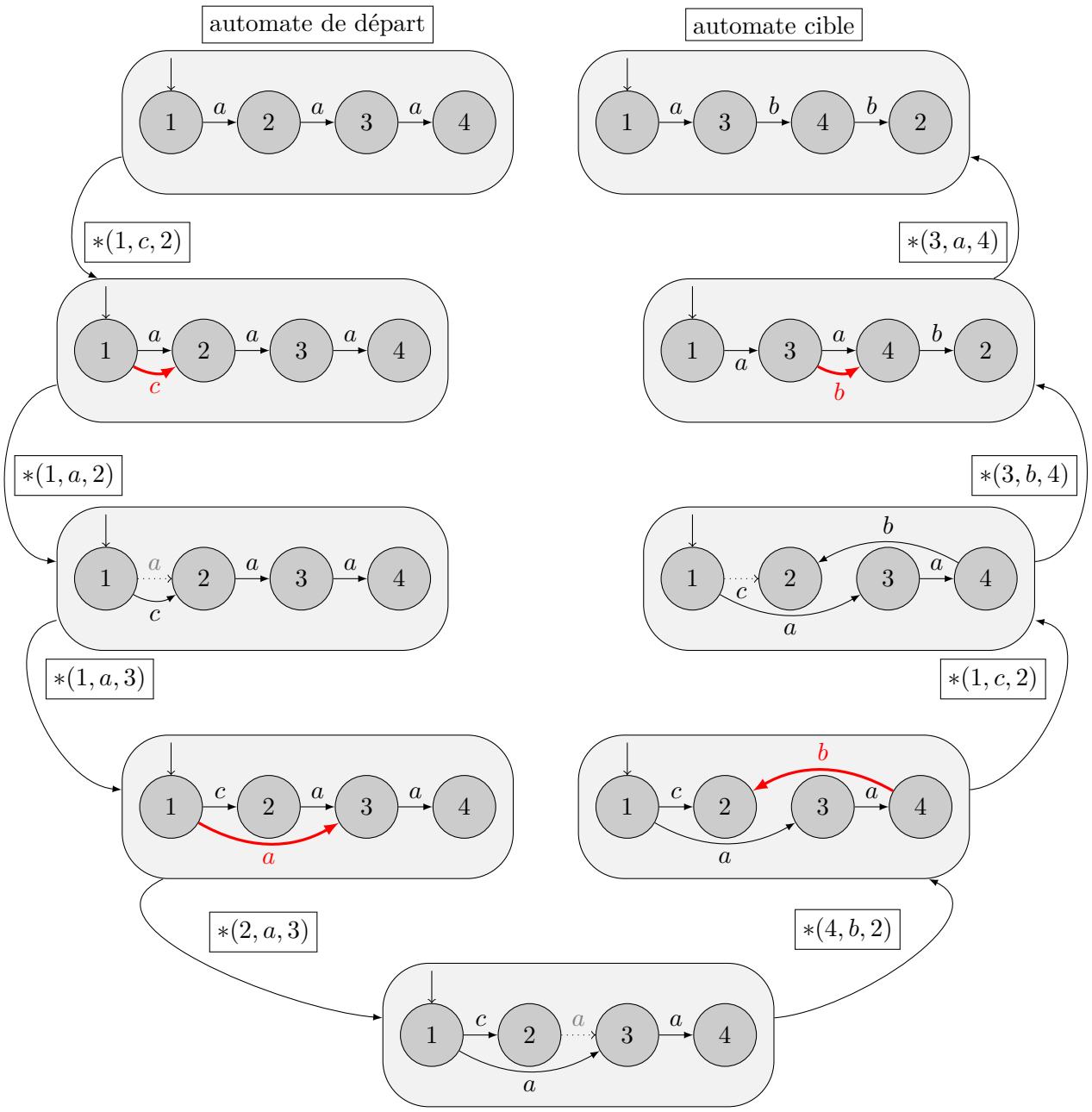


FIGURE 6.4 – Déplacement d'un automate sérié à un autre

Lemme 62 (Chemin d'un automate à un automate sérié)**.** *On considère $x \in \mathbb{P}_n$, il existe un chemin dans G de x à un automate sérié de $\mathbb{P}_n$.*

Preuve. On considère $x \in \mathbb{P}_n$, il existe alors un chemin dans G de x à son arbre des sources $\sigma(x)$. En effet, $\sigma(x)$ est obtenu par des suppressions de transitions qui correspondent à des mouvements possibles dans le graphe G.

Si l'arbre des sources $\sigma(x) = (\llbracket 1, n \rrbracket, \Sigma, E_\sigma, \{1\})$ obtenu n'est pas sérié, on peut considérer un état p de $\sigma(x)$ tel que p admette au moins deux transitions sortantes pointant vers deux états q_1 et q_2 distincts.

Il existe alors $a \in \Sigma$ tel que $(p, a, q_2) \in E_\sigma$.

On considère alors f une feuille de $\sigma(x)$ telle que q_1 soit un de ses ancêtres, ou telle que $f = q_1$ si q_1 est une feuille de $\sigma(x)$. Alors en posant $z \stackrel{\text{def}}{=} \sigma(x) * (f, a, q_2) * (p, a, q_2)$, z est un arbre pour lequel p possède un descendant de moins que dans l'arbre $\sigma(x)$.

On répète éventuellement le même procédé jusqu'à obtenir un automate sérié. □

Le principe du passage de l'arbre des sources à un automate sérié est illustré par la figure 6.5.

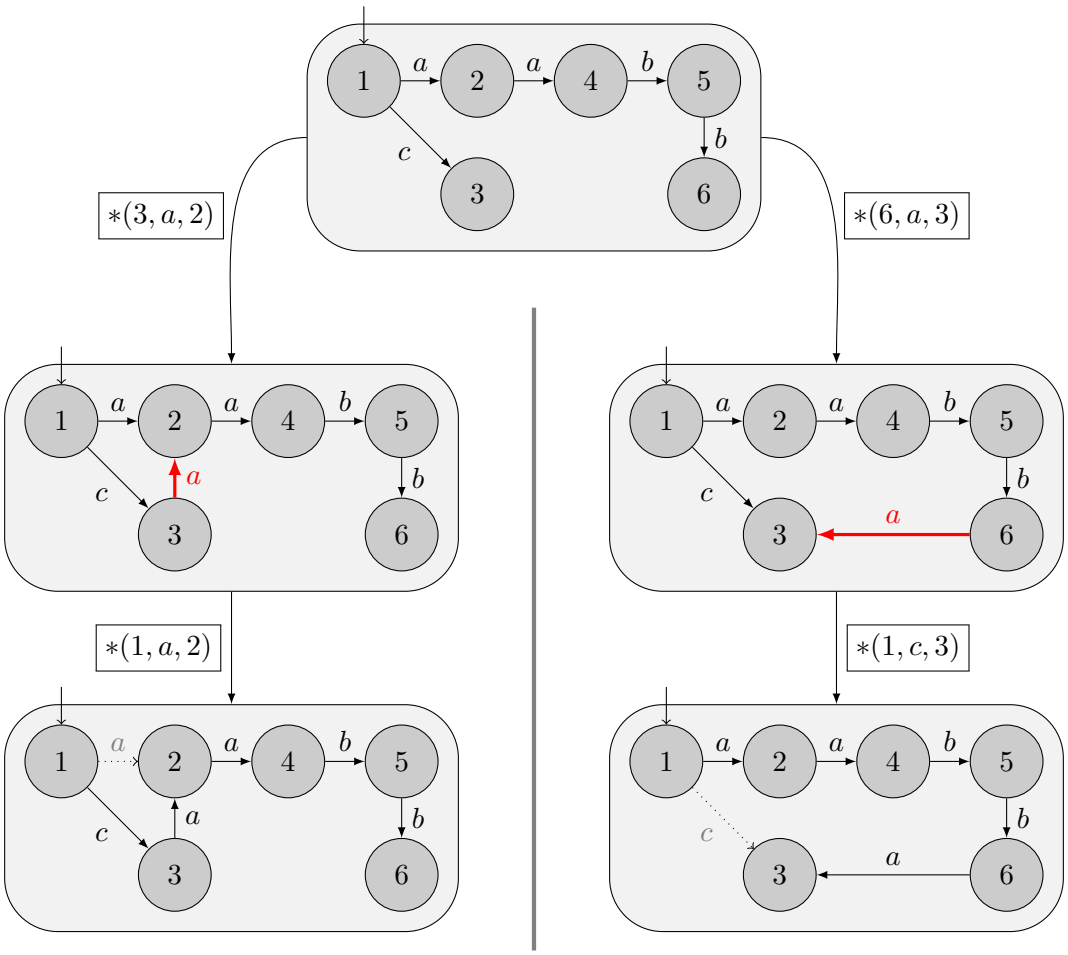


FIGURE 6.5 – Exemples de passage de l'arbre des sources à un automate sérié. On donne deux chemins possibles correspondant aux solutions décrites dans le lemme 62

On peut maintenant établir le résultat essentiel de la proposition 63.

Proposition 63 (irréductibilité)**.** *La chaîne de Markov associée à M est irréductible.*

Preuve. On considère $(x, y) \in \mathbb{P}_n^2$ un couple d'automates, le résultat du lemme 62 assure l'existence d'un chemin de x à un automate sérié $s(x)$ ainsi qu'un chemin de y à un automate sérié $s(y)$.

Le résultat du lemme 58 assure que le chemin de y à $s(y)$ est réversible (on peut le prendre dans le sens inverse).

Le résultat du lemme 61 permet lui d'établir l'existence d'un chemin de $s(x)$ à $s(y)$.

Finalement il est possible de relier x à y dans G, la chaîne est donc irréductible. □

Les démonstrations précédentes permettent d'établir un résultat intéressant sur le diamètre du graphe G associé à la chaîne.

Proposition 64 (diamètre)**.** *Le diamètre* $\mathrm{diam}(G)$ *de* G *vérifie* $\mathrm{diam}(G) = \Theta(n)$.

Preuve. La distance dans G entre deux automates distincts sériés est au moins égal à $n-1$ si toutes les transitions sont différentes. On obtient ainsi la minoration souhaitée.

La distance entre un automate x et l'arbre $\sigma(x)$ associé est majorée par $|\Sigma|n$.

La preuve du lemme 62 permet d'affirmer que la distance de $\sigma(x)$ à un automate sérié est majorée par $2(n-1)$.

De la même façon, la preuve du lemme 61, assure que la distance entre deux automates sériés distincts est majorée par $6n$.

On peut donc majorer la distance entre deux automates par $2(|\Sigma|+5)n$. □

On parvient ainsi au résultat qui valide la construction de la chaîne en vue de la mise en place d'une méthode de Monte-Carlo par chaîne de Markov.

Proposition 65 (ergodicité, symétrie, distribution uniforme)**.** *La chaîne de Markov associée à* M *est ergodique et symétrique, sa distribution stationnaire est donc la distribution uniforme sur* $\mathbb{P}_n$.

Preuve. On a établi que pour tout $x \in \mathbb{P}_n$, $M(x,x) > 0$, la proposition 63 assure le caractère irréductible de la chaîne qui est donc également apériodique. Enfin le résultat du lemme 59 qui garantit que la chaîne est symétrique permet de conclure à propos de la distribution stationnaire. □

6.2.2 Génération aléatoire d'automates partiellement ordonnés avec au plus m boucles

Les méthodes de Monte-Carlo par chaîne de Markov présentent l'inconvénient de ne pas donner un accès aisé au nombre de pas nécessaires dans la chaîne pour garantir un générateur proche de la distribution stationnaire. En clair, l'encadrement du temps de mélange présente souvent des difficultés. Un des aspects de notre travail consiste justement à montrer comment des méthodes statistiques peuvent valider un générateur pour lequel un encadrement du temps de mélange est trop imprécis. Dans cette section nous illustrons l'idée que si la validation théorique de l'outil construit est difficile, il est en revanche aisé de l'adapter à la génération d'autres classes d'objets.

On considère donc dans cette cette section la sous-classe $\mathbb{P}_n^{[\leq m]}$ des automates partiellement ordonnés à n états $\mathbb{P}_n$ pour lesquels au plus m transitions sont des boucles.

Comme précédemment, le générateur ne dotera pas les automates d'états finals. Ceux-ci seront tirés *a posteriori*.

On considère $x \in \mathbb{P}_n^{[\leq m]}$, un automate partiellement ordonné possédant au plus m transitions qui sont des boucles. Les transitions dans la chaîne seront définies par le tirage aléatoire d'un triplet (p,a,q) de $Q \times \Sigma \times Q$, en examinant les cas suivants :

- Le triplet (p,a,q) est une transition de x. On considère alors y l'automate obtenu à partir de x en supprimant la transition (p,a,q). Si $y \in \mathbb{P}_n^{[\leq m]}$ alors $x \circ (p,a,q)$ est défini par $x \circ (p,a,q) \stackrel{\text{def}}{=} y$. Si $y \notin \mathbb{P}_n^{[\leq m]}$ alors $x \circ (p,a,q)$ est défini par $x \circ (p,a,q) \stackrel{\text{def}}{=} x$.

- Le triplet (p, a, q) n'est pas une transition de x et par ailleurs, il n'existe aucune transition de la forme (p, a, r) dans l'automate x où $r \in [\![1, n]\!]$. On considère alors l'automate y obtenu à partir de x en ajoutant la transition (p, a, q). Comme précédemment, si $y \in \mathbb{P}_n^{[\leq m]}$ alors $x \circ (p, a, q)$ est défini par $x \circ (p, a, q) \stackrel{\text{def}}{=} y$. Si $y \notin \mathbb{P}_n^{[\leq m]}$ alors $x \circ (p, a, q)$ est défini par $x \circ (p, a, q) \stackrel{\text{def}}{=} x$.
- Le (p, a, q) n'est pas une transition de x et il existe par ailleurs, dans l'automate x, une transition de la (p, a, r), avec $r \in [\![1, n]\!] \backslash \{q\}$. On considère alors l'automate y obtenu à partir de x en supprimant la transition (p, a, r) et en la ajoutant la transition (p, a, q). Comme précédemment, si $y \in \mathbb{P}_n^{[\leq m]}$ alors $x \circ (p, a, q)$ est défini par $x \circ (p, a, q) \stackrel{\text{def}}{=} y$. Si $y \notin \mathbb{P}_n^{[\leq m]}$ alors $x \circ (p, a, q)$ est défini par $x \circ (p, a, q) \stackrel{\text{def}}{=} x$.

La construction de la matrice de transition $M^{[\leq m]}$ sur l'ensemble $\mathbb{P}_n^{[\leq m]}$, s'appuie, comme dans la section précédente, sur la définition précédente de l'opérateur $\circ$.

Pour tout couple d'automates **distincts** (x, y) de $\mathbb{P}_n^{[\leq m]}$, s'il existe un triplet (p, a, q) tel que $y = x \circ (p, a, q)$, on pose :

$$M^{[\leq m]}(x, y) \stackrel{\text{def}}{=} \frac{1}{|\Sigma| n^2}.$$

Dans le cas contraire, toujours avec $x \neq y$, si pour tout triplet (p, a, q), $y \neq x \circ (p, a, q)$, on pose $M^{[\leq m]}(x, y) \stackrel{\text{def}}{=} 0$.

Ici encore, afin que la matrice $M^{[\leq m]}$ soit stochastique, la définition de $M^{[\leq m]}(x, x)$ est contrainte :

$$M^{[\leq m]}(x, x) \stackrel{\text{def}}{=} 1 - \sum_{y \in \mathbb{P}_n^{[\leq m]} \backslash \{x\}} M^{[\leq m]}(x, y).$$

Puisque l'on peut considérer $|\Sigma| n^2$ triplets il existe au plus $|\Sigma| n^2$ automates y différents de x tels que $M^{[\leq m]}(x, y) \neq 0$. Puisque tous les coefficients non-nuls et non diagonaux de la matrice $M^{[\leq m]}$ ont la même valeur $\frac{1}{|\Sigma| n^2}$, le coefficient diagonal $M^{[\leq m]}(x, x)$ est donc positif.

Le même argument que celui employé dans le cas général à la page 115 permet de garantir que $M^{[\leq m]}(x, x) > 0$.

On peut établir des résultats comparables à ceux obtenus dans la section précédente.

Lemme 66. *On considère $x \in \mathbb{P}_n^{[\leq m]}$ et $(p, a, q) \in Q \times \Sigma \times Q$. Si $x \circ (p, a, q) \neq x$ alors il existe $q' \in Q$ tel que $(x \circ (p, a, q)) \circ (p, a, q') = x$.*

Preuve. La démonstration est la même que celle du lemme 58. □

Proposition 67 (ergodicité, symétrie, distribution uniforme, $\mathbb{P}_n^{[\leq m]}$)**.** *La chaîne de Markov associée à $M^{[\leq m]}$ est ergodique et symétrique, sa distribution stationnaire est donc la distribution uniforme sur $\mathbb{P}_n^{[\leq m]}$.*

Preuve. La preuve du résultat est essentiellement la même que celle de la proposition 63. Seule l'irréductibilité nécessite quelques éclaircissements.
On considère $x, y \in \mathbb{P}_n^{[\leq m]}$, on note $b(x)$ et $b(y)$ respectivement le nombres de boucles dans les automates x et y respectivement. Alors $\max(b(x), b(y)) \leqslant m$. On peut considérer le chemin de x à y décrit dans la proposition 63, x et y sont en effet des automates particuliers de $\mathbb{P}_n$. Pour tout automate z de ce chemin on a $b(z) \leqslant \max(b(x), b(y))$, le chemin relie donc des sommets de $\mathbb{P}_n^{[\leq m]}$. La chaîne est donc irréductible. Les autres résultats s'obtiennent comme dans le cas général. □

6.3 Les tests statistiques

Comme exposé dans la partie préliminaire consacrée aux statistiques, les tests que l'on peut mettre en œuvre afin de valider un générateur dépendent de l'ordre de grandeur du cardinal des classes que l'on souhaite engendrer. Encore une fois cette question se résume à "quel sens peut on donner à la génération aléatoire uniforme d'éléments d'une classe de 10^{40} éléments ?".

Nous présenterons donc dans cette section différents tests en distinguant ceux qui sont adaptés à des combinatoires "raisonnables" des tests adaptés à des ensembles dont la taille ne permet pas d'envisager la production de la liste exhaustive de tous les éléments.

6.3.1 Deux tests simples pour un petit nombre d'états

Si le nombre d'états des automates de $\mathbb{P}_n$ est suffisamment faible il est possible d'envisager de produire des échantillons dont la taille est très supérieure à $|\mathbb{P}_n|$. Dans ces cas il est possible de mettre en place des tests statistiques très simple afin de mesurer la convergence de la méthode de Monte-Carlo par chaîne de Markov vers la distribution stationnaire (qui est uniforme ici).

Le test du collectionneur de vignettes

Le résultat classique du *collectionneur de vignettes* donne le nombre moyen μ de tirages nécessaires pour obtenir tous les éléments d'un ensemble de m éléments en supposant qu'à chaque tirage chacun des m éléments a la même probabilité d'apparition [LPW09,].

$$\mu = m \sum_{k=1}^{m} \frac{1}{k}.$$

Quand la combinatoire s'y prête, on peut donc mettre en place un test très simple qui consiste à compter le nombre de tirages effectués pour obtenir les m objets et à le comparer à μ.

La figure 6.6 fournit les résultats pour des automates à 3 états sur un alphabet de 2 lettres.

La famille compte 312 automates, on a donc $\lfloor \mu \rfloor = 1972$.

Pour chaque valeur de t donnant le nombre de pas dans la chaîne, 50 expériences ont été menées et c'est le nombre moyen de tirages nécessaire pour obtenir les 312 automates, qui apparaît sur le graphique. Pour $t \geqslant 100$, les résultats apparaissent très corrects.

Test d'entropie

On note e la taille d'un échantillon engendré par le générateur. Si le générateur est parfait, chaque automate doit apparaître avec la même fréquence $\frac{e}{|\mathbb{P}_n|}$.

En notant m le cardinal de $\mathbb{P}_n$, l'entropie entropie(e) de l'échantillon est définie par :

$$\text{entropie}(e) \stackrel{\text{def}}{=} - \sum_{i \in [\![1,m]\!]} f_i \log_m (f_i),$$

où f_i désigne la fréquence d'apparition de l'automate indicé par i.

Pour un générateur parfait, puisque $f_i = \frac{1}{m}$, on a :

$$\text{entropie}(e) = 1.$$

Pour des automates partiellement ordonnés à 3 états sur un alphabet de 2 lettres, on obtient les résultats du tableau 6.2.

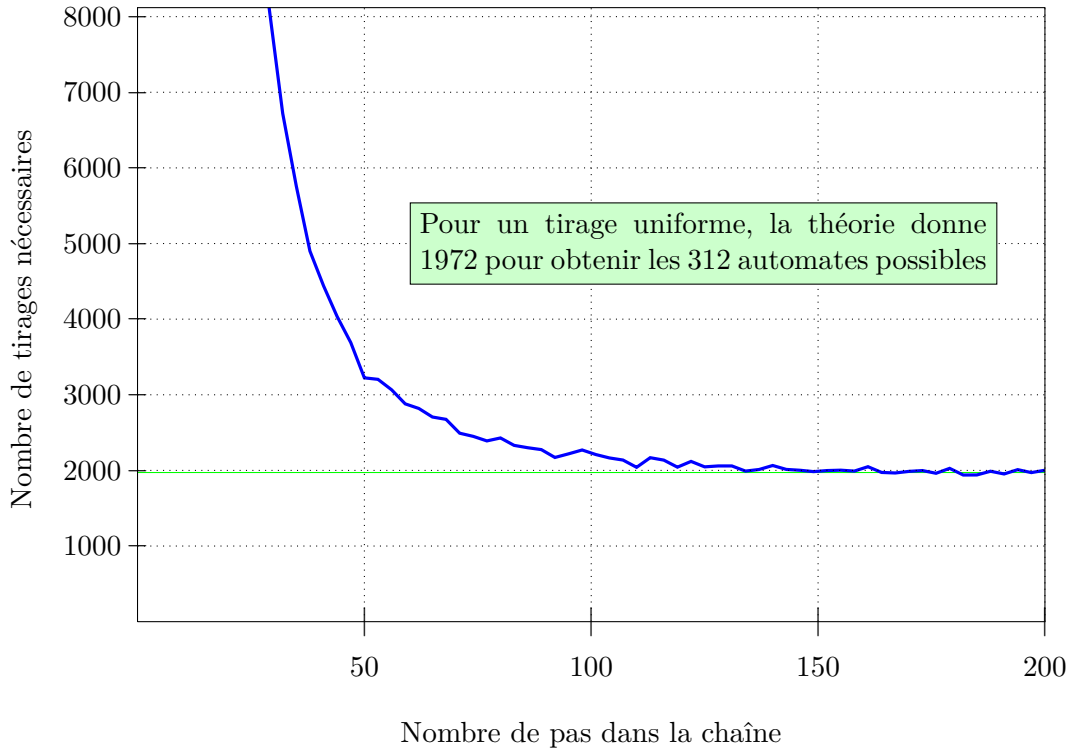


FIGURE 6.6 – Test du collectionneur de vignettes. La courbe montre le nombre moyen de tirages nécessaires pour obtenir les 312 automates à 3 états sur un alphabet à 2 lettres en fonction du nombre de pas effectués dans la chaîne. La moyenne est faite sur 100 tests pour une valeur du nombre de pas dans la chaîne

t	10	50	100	200
entropie(e)	0.88	0.97	0.985	0.987

TABLE 6.2 – Tests d'entropie pour des automates à 3 états sur un alphabet de 2 lettres

On obtient des résultats qui confirment les tests qualitatifs de collectionneur de vignettes.

Ces tests sont en revanche impossibles à mettre en œuvre lorsqu'il est impossible d'envisager une production exhaustive de tous les automates de $\mathbb{P}_n$, or cette impossibilité survient très vite puisque la combinatoire des $\mathbb{P}_n$ est assez explosive !

Pour un nombre d'états plus conséquent il faut donc envisager d'autres tests.

6.3.2 Test pour de grandes échelles : Autocorrélation et Gelman-Rubin

Les tests d'autocorrélation et de Gelman-Rubin sont assez aisés à mettre en place grâce au logiciel **R** [R D08]. Les données d'entrée sont obtenues en itérant les chaînes à partir de 4 points de départ différents.
Si le test de Gelman-Rubin impose au moins deux points de départ différents et "dispersés" dans le graphe associé à la chaîne, cette contrainte n'existe pas dans le test d'autocorrélation. On s'appuiera néanmoins sur la variété des résultats obtenus afin de faire apparaître l'influence éventuelle du point de départ sur les tests d'autocorrélation.

Puisqu'il s'agit de produire des données numériques, on s'attache à observer deux mesures numériques sur les automates obtenus par itérations dans la chaîne :

- Le nombre de transitions de l'automate. Dans les tableaux de résultats cette donnée

est référencée par **mv1**.
- Le nombre de boucles de l'automate. Dans les tableaux de résultats cette donnée est référencée par **mv2**.

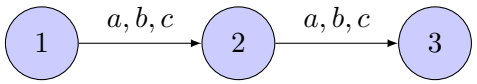
Il est clair que ces choix sont arbitraires et que d'autres mesures faciles d'accès sont également exploitables. On peut par exemple penser à la plus longue chaîne dans l'automate.

On rappelle ici que les 4 automates de départ sont les suivants :

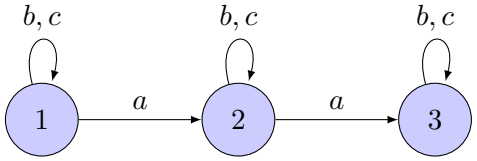
ligne, boucle, arbre, creux

Pour un automate à n états sur un alphabet Σ les différents types sont définis de la façon suivante :

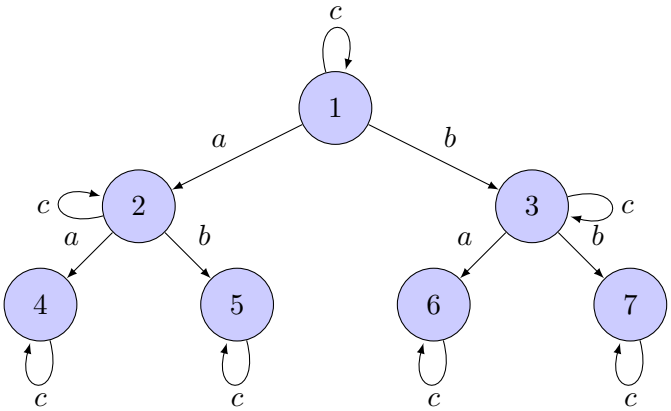
- **ligne**
  En supposant les états numérotés de 1 à n, l'ensemble des transitions est constitué des transitions de la forme $i \xrightarrow{\alpha} i+1$ pour $1 \leqslant i < n$.
  Pour $n = 3$ et $\Sigma = \{a, b, c\}$, par exemple, l'automate de départ est donc :



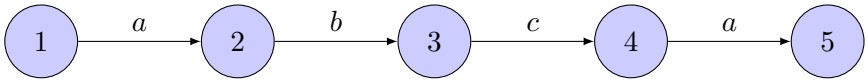
- **boucle**
  En supposant les états numérotés de 1 à n, l'ensemble des transitions est constitué des transitions de la forme $i \xrightarrow{a} i+1$ pour $1 \leqslant i < n$, où a est la première lettre de l'alphabet Σ ainsi que des boucle $i \xrightarrow{\alpha} i$ pour toutes les autres lettres de l'alphabet.
  Pour $n = 3$ et $\Sigma = \{a, b, c\}$, par exemple, l'automate de départ est donc :



- **arbre**
  En supposant les états numérotés de 1 à n, le graphe sous-jacent est un arbre binaire obtenu comme un tas. Les transitions vers un état pair sont étiquetées par la première lettre de l'alphabet Σ, les transitions vers un état impair sont étiquetées par la deuxième lettre de l'alphabet. Si il existe une troisième lettre dans l'alphabet Σ, on définit sur chaque état une boucle étiquetée par cette troisième lettre.
  Pour $n = 6$ et $\Sigma = \{a, b, c\}$, par exemple, l'automate de départ est donc :



- **creux**
  En supposant les états numérotés de 1 à n, l'ensemble des transitions est constitué des transitions de la forme $i \xrightarrow{a_i} i+1$ pour $1 \leqslant i < n$, où a_i est la i-ième (modulo la longueur de l'alphabet Σ) lettre de l'alphabet Σ. Pour $n = 5$ et $\Sigma = \{a, b, c\}$, par exemple, l'automate de départ est donc :



Distances entre les 4 types d'automates de départ

Pour donner du sens aux tests de Gelman-Rubin il faut que les différents points de départ dans le graphe associé à la chaîne soient aussi éloignés que possible.
On a démontré que le diamètre de la chaîne est majoré par $2\,(|\Sigma|+5)\,n$. Une distance en $\theta(n)$ sera donc satisfaisante.

- Pour passer du type ligne au type boucle, il est clair qu'il faut modifier $|\Sigma-1|$ transitions pour $n-1$ états, la distance entre ces deux automates est donc $|\Sigma-1|(n-1)$ au minimum et il est clair que cette valeur est aussi la distance entre les deux automates.
- Pour passer du type ligne au type arbre toutes les transitions, à l'exception de $(1,a,2)$, de l'automate de départ doivent être modifiées. La distance entre les deux automates de départ est donc au moins égales à $|\Sigma|(n-1)-1$.
- Pour passer du type ligne au type creux il faut supprimer $|\Sigma-1|(n-1)$ transitions dans l'automate de départ. Cet entier est clairement la distance entre les deux automates.
- Pour passer du type boucle au type arbre il faut modifier au moins $2(n-1)-1$ transitions dans l'automate de départ. La distance entre les deux automates est donc minorée par cette valeur.
- Pour passer du type boucle au type creux il faut supprimer au moins $|\Sigma-1|(n-1)$ boucles dans l'automate de départ (et en modifier d'autres). La distance entre les deux automates est donc minorée par cette valeur.
- Enfin, pour passer du type arbre au type creux il faut supprimer modifier au moins $|\Sigma-1|(n-1)$ transitions dans l'automate. Cet entier est clairement la distance entre les deux automates.

Cette rapide étude valide les choix qui ont été faits quant au point de départ des chaînes dans le graphe.

Autocorrélation

Les figures 6.7, 6.8 et 6.9 (voir les annexes en section 6.5), présentent les résultats des tests d'autocorrélation pour des automates à 10, 50 et 100 états sur un alphabet de 3 lettres. Les coefficients sont présentés pour les 4 points de départ qui seront utilisés dans les tests de Gelman-Rubin.

Les résultats des tests montrent que les coefficients d'autocorrélation sont décroissants. On note tout de même une influence assez nette du point de départ dans la chaîne. Il faudrait par ailleurs lancer d'autres tests afin de mesurer l'influence du critère numérique retenu.

On peut tout de même conclure que ces tests ne permettent pas d'invalider notre hypothèse sur le temps de mélange.

Gelman-Rubin

Les figures 6.10, 6.11, 6.12, 6.13, 6.14 et 6.15 présentent en détails les résultats obtenus pour des automates à 10 états sur un alphabet de 3 lettres pour différentes temps d'itération.

On rappelle ici que le test a consisté à lancer 4 chaînes avec des points de départ différents : les automates *ligne*, *boucle*, *arbre* et *creux* décrits plus haut. On rappelle aussi

que deux paramètres numériques sont testés : le nombre de transitions et le nombre de boucles, correspondant à **mv1** et **mv2** respectivement dans les résultats présentés.

Le calcul est mené à l'aide du logiciel `R` [R D08] qui prend en compte certaines améliorations suggérées dans [BG98]. En particulier le *potentiel scale reduction factor* est multiplié par un facteur ν qui prend en compte la variabilité des variances estimées par les échantillons. L'hypothèse consiste à dire que les quantités estimées (le nombre de transitions et le nombre de boucles dans notre expérience) suivent une loi de t de Student à d degré de liberté. Soit encore la loi d'une variable aléatoire de la forme :

$$\frac{Z}{\sqrt{U/d}},$$

où les variables aléatoires Z et U sont indépendantes. La variable aléatoire Z suit une loi normale centrée réduite, U une loi du χ^2 à d degrés de liberté. La valeur de d est évaluée par la méthode dite *des moments*. Le facteur multiplicatif est défini par :

$$\nu = \sqrt{\frac{d+1}{d+3}}$$

Le résultat est le `shrink factor` qui apparaît dans les tableaux de résultats fournis par le logiciel. Pour affiner la lecture du graphique, les *shrink factors* sont calculés sur des morceaux de chaîne de la forme $[1 : 50 \cdot k]$. L'indice 1 fait référence au premier sommet dont on tient compte, la variable k est un entier. L'indication `last iteration in chain` fait référence à l'indice du dernier sommet pour lequel les paramètres numériques ont été calculés. Les courbes continue et pointillée qui apparaissent sur les graphiques font respectivement référence à la moyenne et au 0.975-quantile supérieur.

Le test de Gelman-Rubin présente l'avantage de pouvoir être mis en place pour des familles d'automates comptant de nombreux représentants. Les figures de 6.16 à 6.21 (voir les annexes en section 6.5) présentent en détails les résultats obtenus pour des automates à 50 états et des automates à 100 états sur un alphabet de 3 lettres.

Dans ce cas encore, les tests ne permettent pas d'invalider notre hypothèse sur les temps de mélange. On peut par ailleurs lancer de façon raisonnable le test pour des automates à plusieurs centaines d'états. Le *potential scale reduction factor* tend vers 1. Les quatre points de départ dispersés dans la chaîne renforcent la confiance que l'on peut faire dans la construction de la chaîne qui semble *bien mélanger*.

6.3.3 Test pour de grandes échelles : χ^2

Les générateurs qui s'appuient sur des méthodes de Monte-Carlo par chaîne de Markov ou sur la méthode récursive peuvent fournir des échantillons d'éléments qui appartiennent à des ensembles de cardinaux si grands qu'il est impossible d'envisager la production de tous les éléments de l'ensemble.

Dans le cas qui nous occupe ici, même pour un nombre d'états raisonnables, $|\mathbb{P}_n|$ est très grand. Pour 20 états et un alphabet à deux lettres, $|\mathbb{P}_n|$ est par exemple de l'ordre de 10^{45}.

Une solution pour élaborer un test statistique consiste à partitionner l'ensemble en différentes classes afin de pouvoir réaliser un test du Chi-deux de Pearson.

Puisque les méthodes combinatoires usuelles permettent de déterminer de manière exacte le nombre d'automates ayant un nombre de boucles donné, c'est sur cette idée que l'on construit la partition de $\mathbb{P}_n$. Il faut que la partition obtenue suive la règle de Cochran, à savoir chaque classe doit contenir un minimum d'objets. Pour réaliser cet objectif il suffit

de regrouper les classes qui regroupent un nombre trop faible d'éléments dans une même classe. Par exemple, pour un échantillon de 10 000 automates à 14 états sur un alphabet de 2 lettres, le tableau 6.3 donne les poids théoriques des classes correspondant à un nombre de boucles variant de 9 à 15.

boucles	9	10	11	12	13	14	15
poids	6.68	0.87	0.08	$6 \cdot 10^{-3}$	$2 \cdot 10^{-4}$	$7 \cdot 10^{-6}$	$9 \cdot 10^{-8}$

TABLE 6.3 – Poids des classes de boucles pour des automates à 14 états sur un alphabet à 2-lettres, pour un échantillon de 10 000 automates

Pour respecter la règle de Cochran, on regroupe donc les classes 9 à 15 en une seule classe. On obtient alors le tableau 6.4.

boucles	0	1	2	3	4	5	6	7
poids	190.75	944.86	2057.14	2616.30	2180.80	1266.88	532.06	165.16
boucles	8	$9-15$	$\leftarrow 10$	$\leftarrow 11$	$\leftarrow 12$	$\leftarrow 13$	$\leftarrow 14$	$\leftarrow 15$
poids	38.35	7.65	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$

TABLE 6.4 – La table 6.3 adaptée à la règle de Cochran en regroupant les classes de 9 à 15

Le but consiste ici à évaluer empiriquement le nombre de pas à réaliser dans une méthode de Monte-Carlo par chaîne de Markov pour obtenir un générateur correct.

Il s'agit de tester l'hypothèse H_0 qui consiste à dire que les fréquences des différentes classes définies par le nombre de boucles des automates engendrés, correspondent aux fréquences théoriques calculées dans $\mathbb{P}_n$.

Les résultats de la table 6.5 donnent les χ^2 et les p-valeurs pour un échantillon de 10 000 automates. Le nombre d'automates pour un nombre de boucles fixé est une moyenne faite sur 20 tirages.

Les résultats du tableau 6.3 donne le nombre de classes adapté à la règle de Cochran pour un échantillon de 10 000 automates à 14 états sur un alphabet à 2 lettres. On considère donc 10 classes, pour lesquelles, avec $\alpha = 0.05$, on obtient $t_\alpha = 16.9190$ et $t_\alpha/d \simeq 1.88$.

t	50	100	150	200	250	300	350	400	450	500
χ^2/d	3121.49	632.46	194.1	66.85	25.22	10.12	3.57	1.68	0.6	0.26
p-value	$\simeq 0$	$\simeq 0$	$\simeq 0$	$\simeq 0$	$\simeq 0$	$\simeq 0$	0.0002	0.09	0.79	0.98

TABLE 6.5 – Test du χ^2 pour des échantillons de 10 000 automates partiellement ordonnés de 14 états sur un alphabet à 2 lettres.

Les résultats du tableau 6.5 suggèrent que des valeurs de t vérifiant $t \in [\![400, 450]\!]$ fournissent un bon ordre de grandeur du temps de mélange pour les automates partiellement ordonnés considérés.

Les calculs [2] des résultats du tableau 6.5 conduisent à une stratégie empirique générale destinée à obtenir un ordre de grandeur du temps de mélange pour un générateur construit

2. Les calculs ont été réalisés sur le super-calculateur du Mésocentre de calcul de Franche-Comté.

par méthode de Monte-Carlo par chaîne de Markov pour des automates à nombre d'états fixé, sur un alphabet de taille fixée.

La première idée simple consiste à calculer des valeurs du χ^2 pour des valeurs de t croissantes.
Les premiers résultats peuvent servir de lignes directrices afin de construire des tests un peu plus fins.
Les résultats des tableaux 6.6 , 6.7 et 6.8 proposent des valeurs de χ^2 et des p-valeurs pour un nombre de pas t égal à n^2, n^3 et $10 \cdot n^3$ où n est le nombre d'états.

n	9				11			
$\lvert\Sigma\rvert = 2$	d	t_α/d	χ^2/d	p-val	d	t_α/d	χ^2/d	p-val
$T = n^2$	9	1.88	2700	0	9	1.88	2073	0
$T = n^3$	9	1.88	0.27	0.98	9	1.88	1.85	0.05
$T = 10 \cdot n^3$	9	1.88	0.64	0.76	9	1.88	0.75	0.66

n	13				15			
$\lvert\Sigma\rvert = 2$	d	t_α/d	χ^2/d	p-val	d	t_α/d	χ^2/d	p-val
$T = n^2$	10	1.83	1706	0	10	1.83	1187	0
$T = n^3$	10	1.83	1.84	0.047	10	1.83	0.53	0.86
$T = 10 \cdot n^3$	10	1.83	0.46	0.9	10	1.83	0.93	0.5

n	17				19			
$\lvert\Sigma\rvert = 2$	d	t_α/d	χ^2/d	p-val	d	t_α/d	χ^2/d	p-val
$T = n^2$	11	1.79	914	0	11	1.79	765	0
$T = n^3$	11	1.79	0.89	0.5	11	1.79	0.79	0.64
$T = 10 \cdot n^3$	11	1.79	1.1	0.35	11	1.79		

n	21				23			
$\lvert\Sigma\rvert = 2$	d	t_α/d	χ^2/d	p-val	d	t_α/d	χ^2/d	p-val
$T = n^2$	11	1.79	685	0	11	1.79	580	0
$T = n^3$	11	1.79	0.47	0.91	11	1.79	0.83	0.61
$T = 10 \cdot n^3$	11	1.79			11	1.79		

TABLE 6.6 – Test du χ^2 pour des automates partiellement ordonnés sur un alphabet de 2 lettres, $\alpha = 0.05$, échantillon de $2 \cdot 10^5$ automates

Les résultats des tableaux 6.7 et 6.8 proposent les mêmes mesures mais pour un nombre de lettres différent.

Ces résultats sont assez parcellaires dans la mesure où il nous faudrait pousser les expériences pour un nombre d'états beaucoup plus important pour mettre en valeur le rôle des constantes multiplicatives.

6.4 Conclusions et perspectives

Comme nous l'avons déjà précisé, les tests statistiques que nous avons proposés ne constituent pas une preuve de l'hypothèse faite sur le temps de mélange.
Il faut garder à l'esprit que les techniques d'évaluation des temps de mélange dans les chaînes de Markov, si elles sont variées, sont souvent très difficiles à mettre en œuvre.
Dans le cas des combinatoires imposées par les structures que l'on souhaite engendrer,

n	9				11			
$\|\Sigma\| = 3$	d	t_α/d	χ^2/d	p-val	d	t_α/d	χ^2/d	p-val
$T = n^2$	9	1.88	2700	0	9	1.88	2073	0
$T = n^3$	13	1.72	0.97	0.48	14	1.69	0.46	0.95

n	13				15			
$\|\Sigma\| = 3$	d	t_α/d	χ^2/d	p-val	d	t_α/d	χ^2/d	p-val
$T = n^2$	10	1.83	1706	0	10	1.83	1187	0
$T = n^3$	15	1.66	0.92	0.54	15	1.66	0.85	0.61

TABLE 6.7 – Test du χ^2 pour des automates partiellement ordonnés sur un alphabet de 3 lettres, $\alpha = 0.05$, échantillon de $2 \cdot 10^5$ automates

n	9				11			
$\|\Sigma\| = 5$	d	t_α/d	χ^2/d	p-val	d	t_α/d	χ^2/d	p-val
$T = n^2$	20	1.57	10^5	0	21	1.55	88315	0
$T = n^3$	20	1.57	3.925	$\simeq 0$	21	1.55	1.46	0.05
$T = 10 \cdot n^3$	20	1.57	1.024	0.36	21	1.55	0.75	0.73

n	13				15			
$\|\Sigma\| = 5$	d	t_α/d	χ^2/d	p-val	d	t_α/d	χ^2/d	p-val
$T = n^2$	22	1.54	72035	0	23	1.53	52843	0
$T = n^3$	22	1.54	0.9	0.5	23	1.53	1.46	0.054
$T = 10 \cdot n^3$	22	1.54	0.79	0.6	23	1.53		

TABLE 6.8 – Test du χ^2 pour des automates partiellement ordonnés sur un alphabet de 5 lettres, $\alpha = 0.05$, échantillon de $2 \cdot 10^5$ automates

il parait inenvisageable, même si cette voie reste explorée dans un cadre plus théorique [Leb14], de se lancer dans des techniques spectrales. Au cours de ce travail, des techniques de couplage, de *strong stationary times* ont été envisagées. Si elles n'ont pas abouti, cela ne signifie pas cependant que ces voies soient fermées.
Les développements proposés dans ce chapitre, donnent donc un cadre qui permet, en attendant des résultats plus solides, de se faire une idée de la qualité de mélange de la chaîne construite. Le fait de disposer d'outils à la fois faciles à mettre en place et qui mesurent le mélange dans la chaîne par des stratégies très différentes doit inciter à ne pas se priver de cette approche.

6.5 Annexes

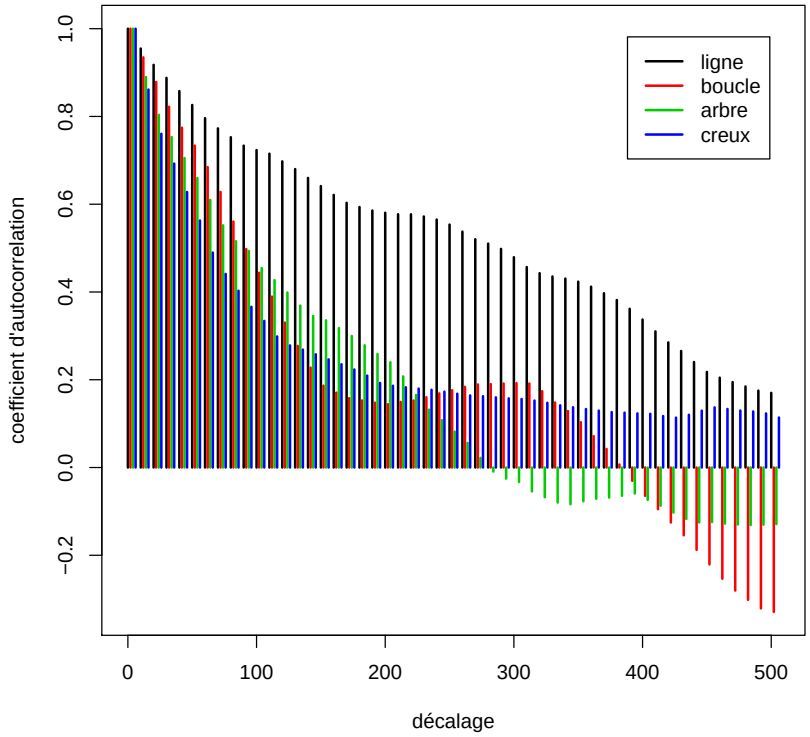


FIGURE 6.7 – Tests d'autocorrélation pour des automates à 10 états sur un alphabet de 3 lettres, pour 2000 itérations

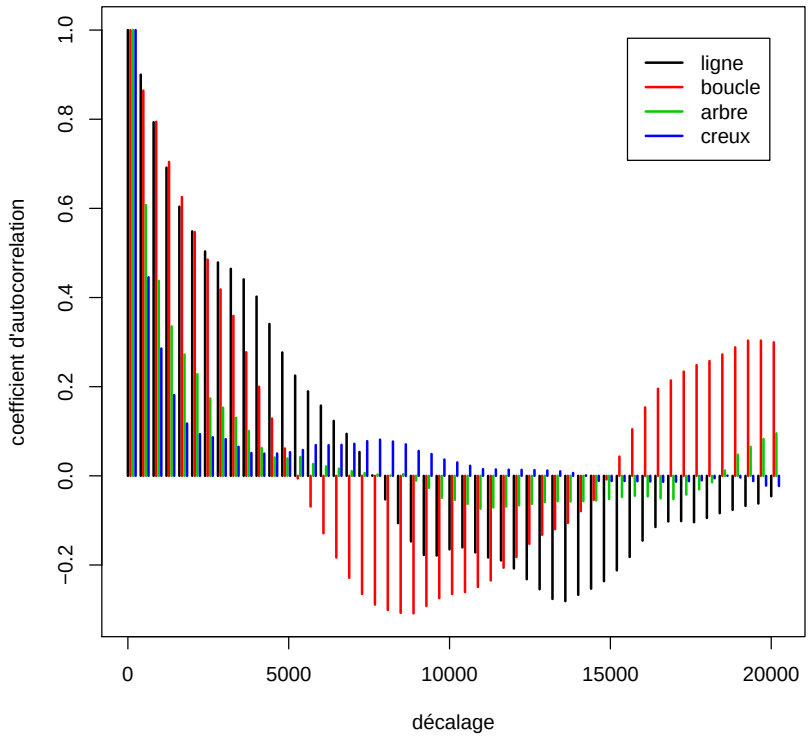


FIGURE 6.8 – Tests d'autocorrélation pour des automates à 50 états sur un alphabet de 3 lettres, pour 100 000 itérations

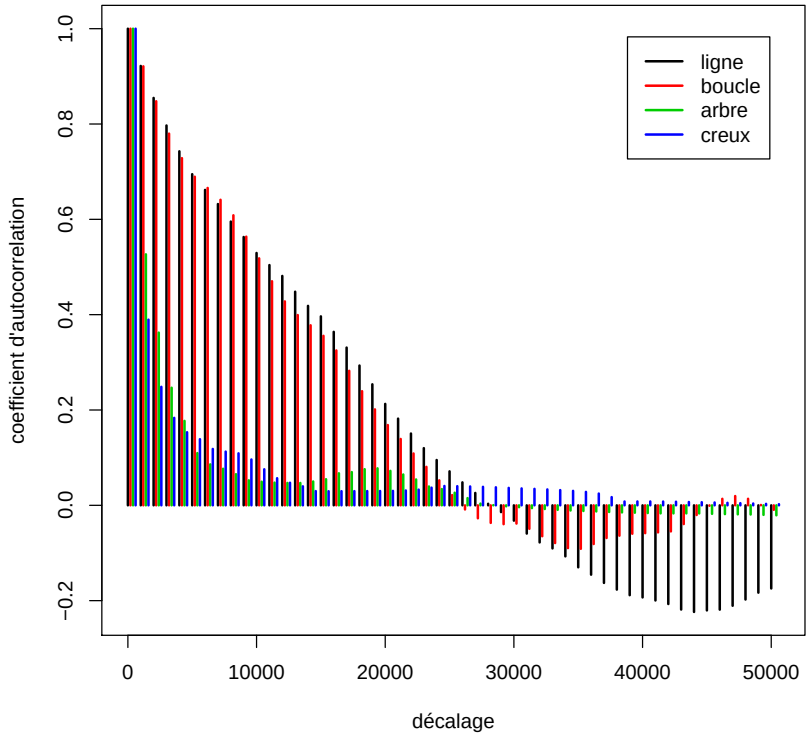


FIGURE 6.9 – Tests d'autocorrélation pour des automates à 100 états sur un alphabet de 3 lettres, pour 500 000 itérations

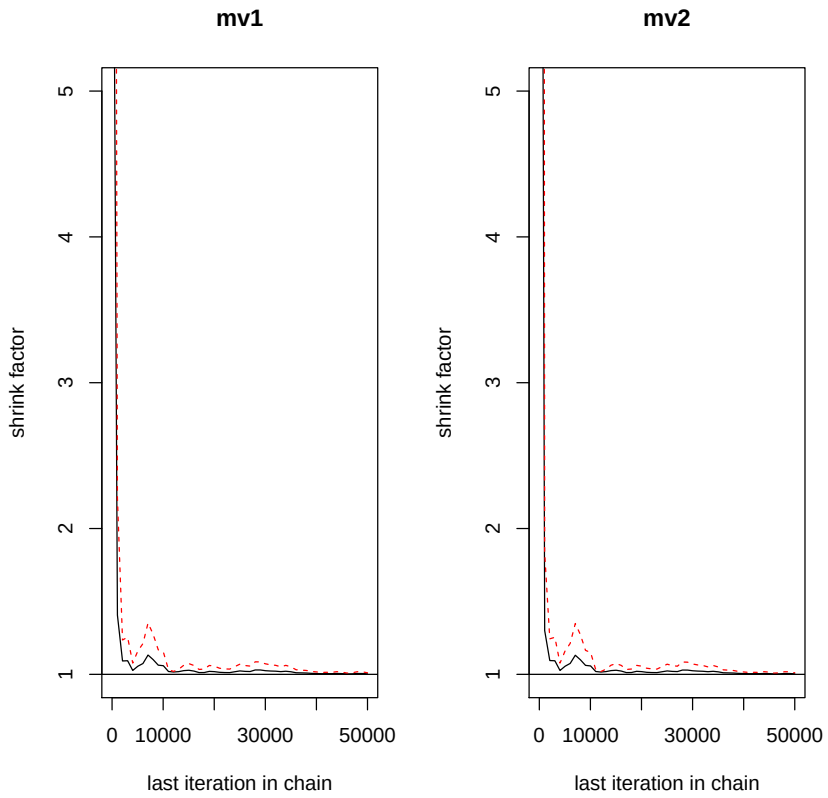


FIGURE 6.10 – Test de Gelman-Rubin pour des automates à 10 états sur un alphabet de 3 lettres, pour 50000 itérations, allure générale des résultats

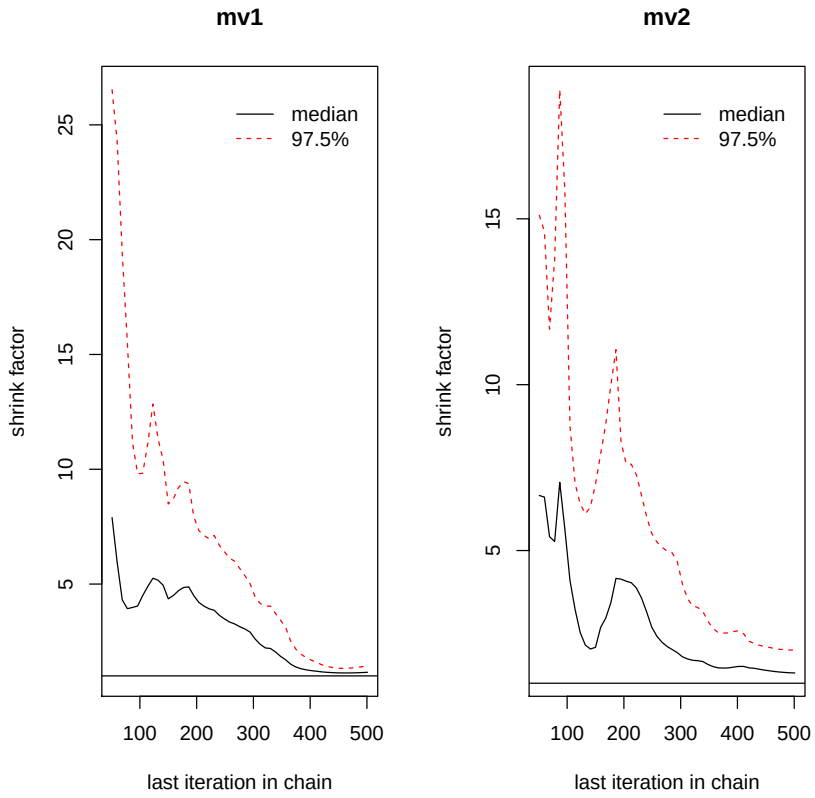


FIGURE 6.11 – Test de Gelman-Rubin pour des automates à 10 états sur un alphabet de 3 lettres, pour 500 itérations

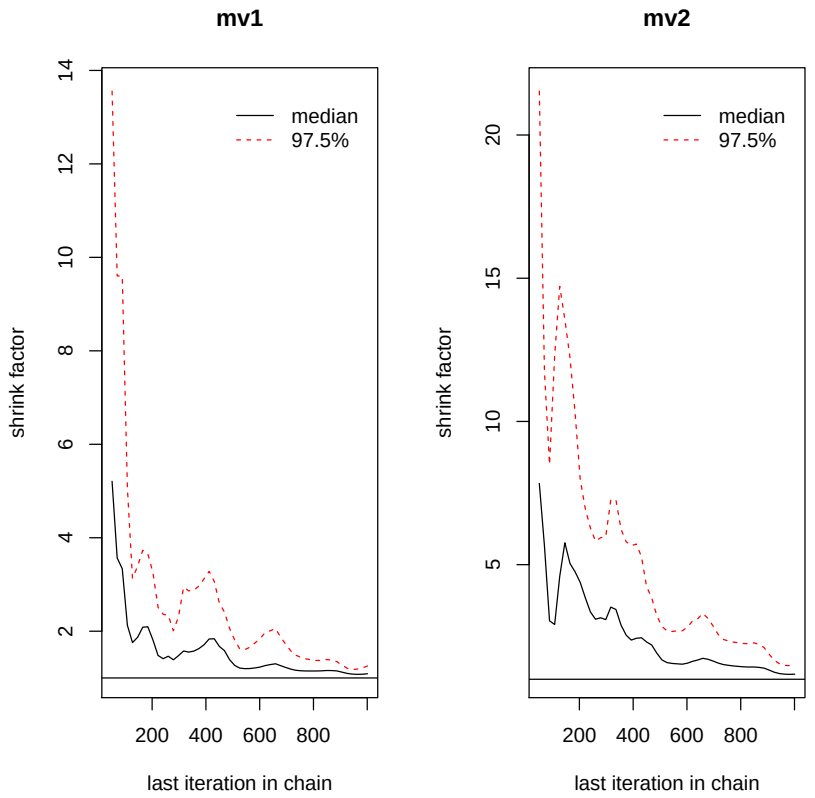


FIGURE 6.12 – Test de Gelman-Rubin pour des automates à 10 états sur un alphabet de 3 lettres, pour 1000 itérations

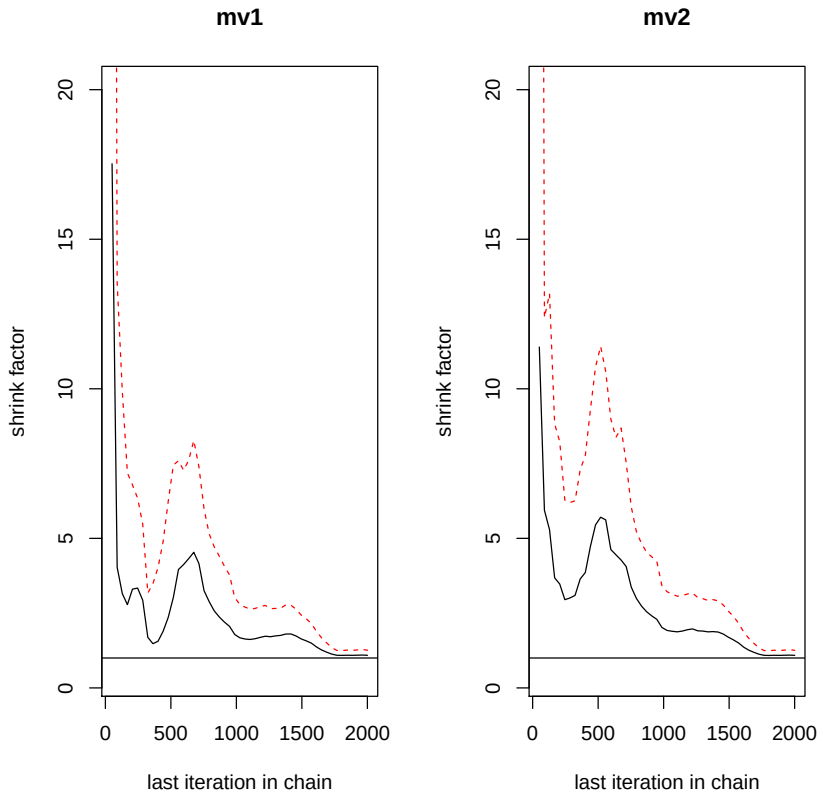


FIGURE 6.13 – Test de Gelman-Rubin pour des automates à 10 états sur un alphabet de 3 lettres, pour 2000 itérations

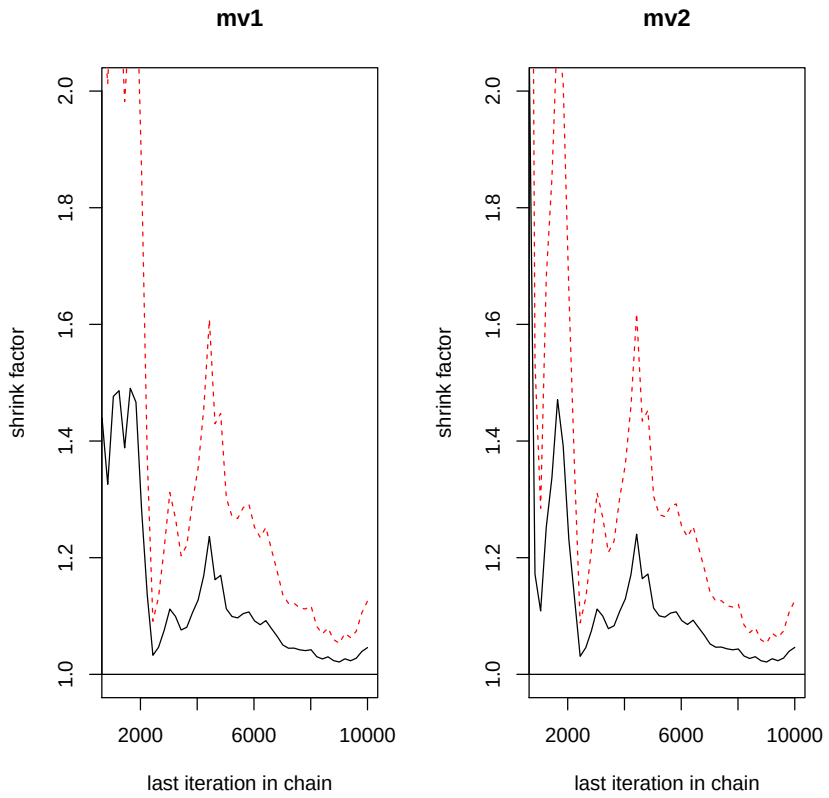


FIGURE 6.14 – Test de Gelman-Rubin pour des automates à 10 états sur un alphabet de 3 lettres, pour 10000 itérations

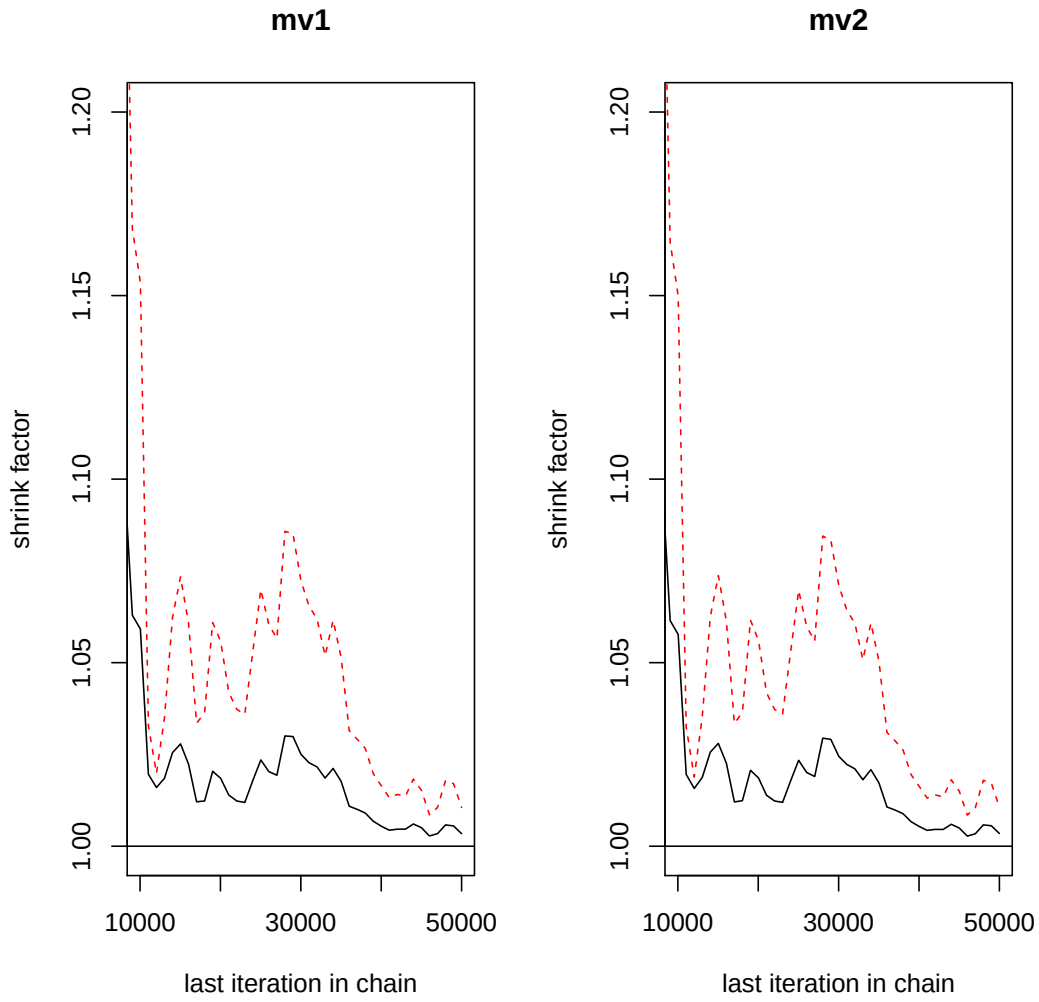


FIGURE 6.15 – Test de Gelman-Rubin pour des automates à 10 états sur un alphabet de 3 lettres, pour 50000 itérations, détails des résultats pour une dernière itération entre 10000 et 50000

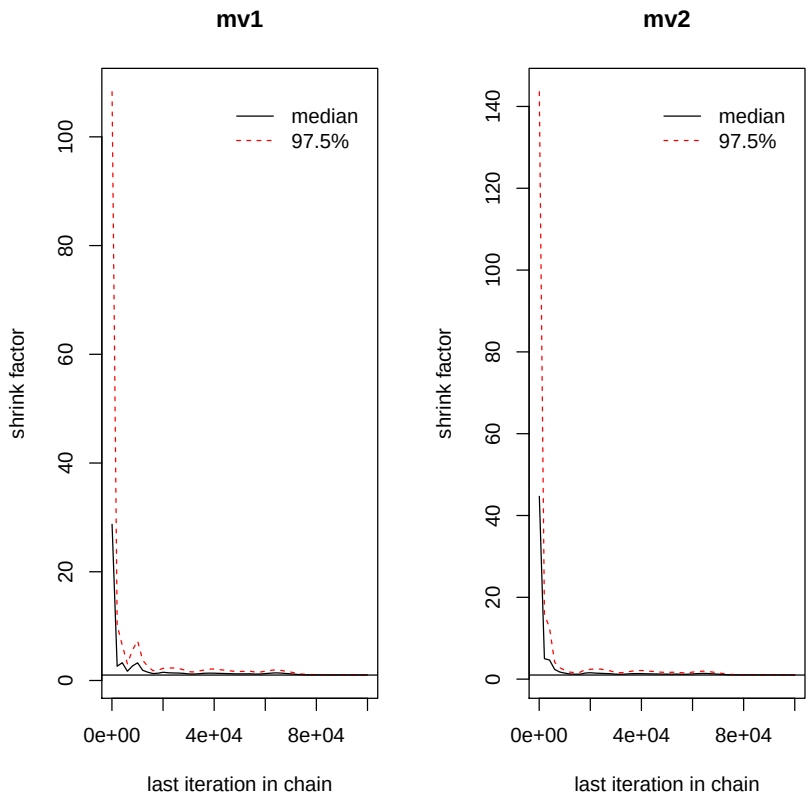


FIGURE 6.16 – Test de Gelman-Rubin pour des automates à 50 états sur un alphabet de 3 lettres, pour 100000 itérations, allure générale des résultats

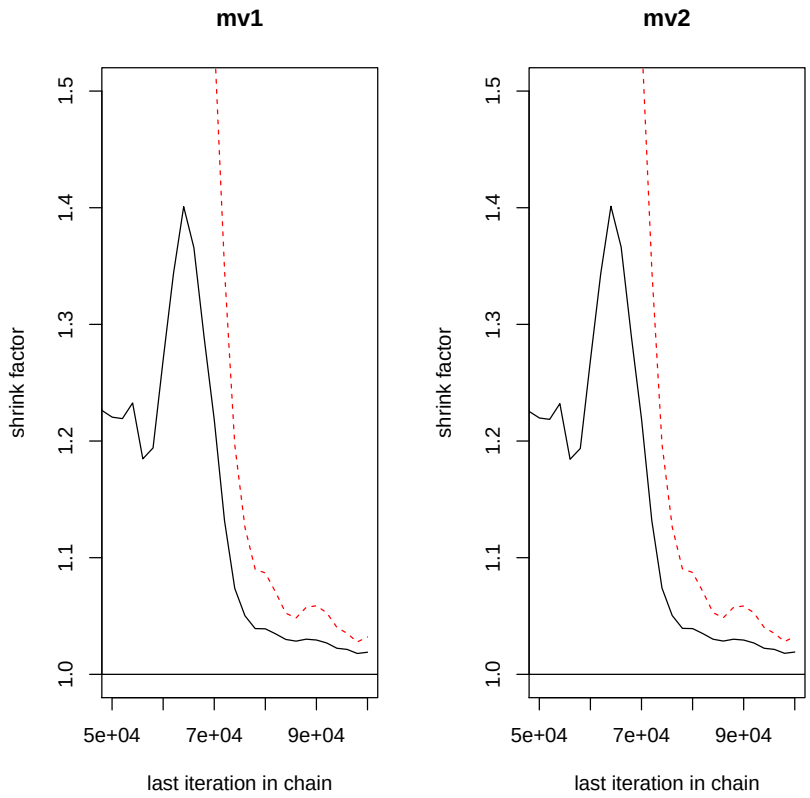


FIGURE 6.17 – Test de Gelman-Rubin pour des automates à 50 états sur un alphabet de 3 lettres, pour 100000 itérations, détails des résultats entre 50 000 et 100 000 itérations

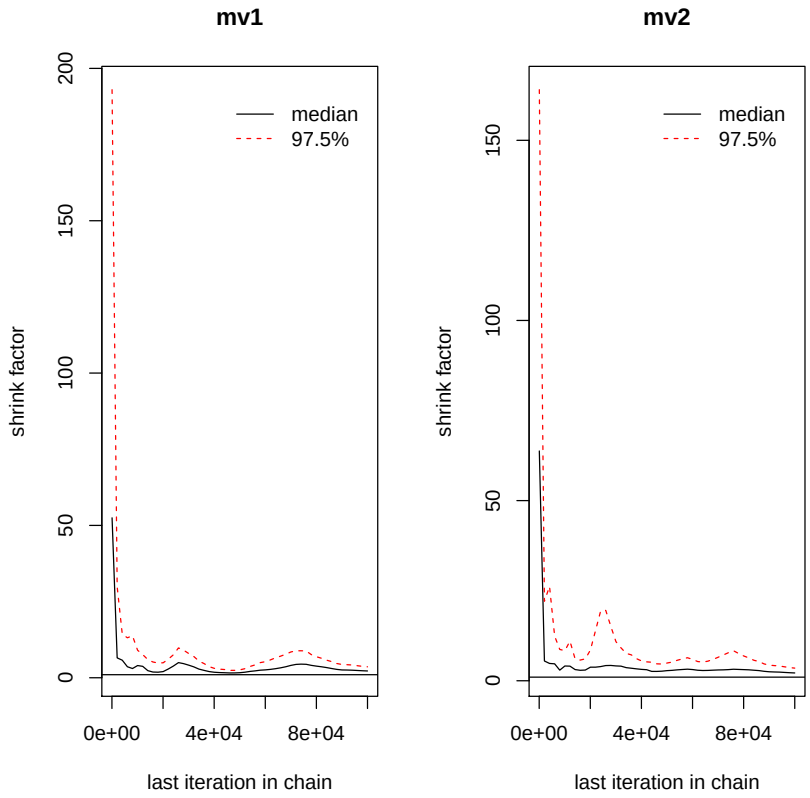


FIGURE 6.18 – Test de Gelman-Rubin pour des automates à 100 états sur un alhabet de 3 lettres, pour 100 000 itérations, allure générale des résultats

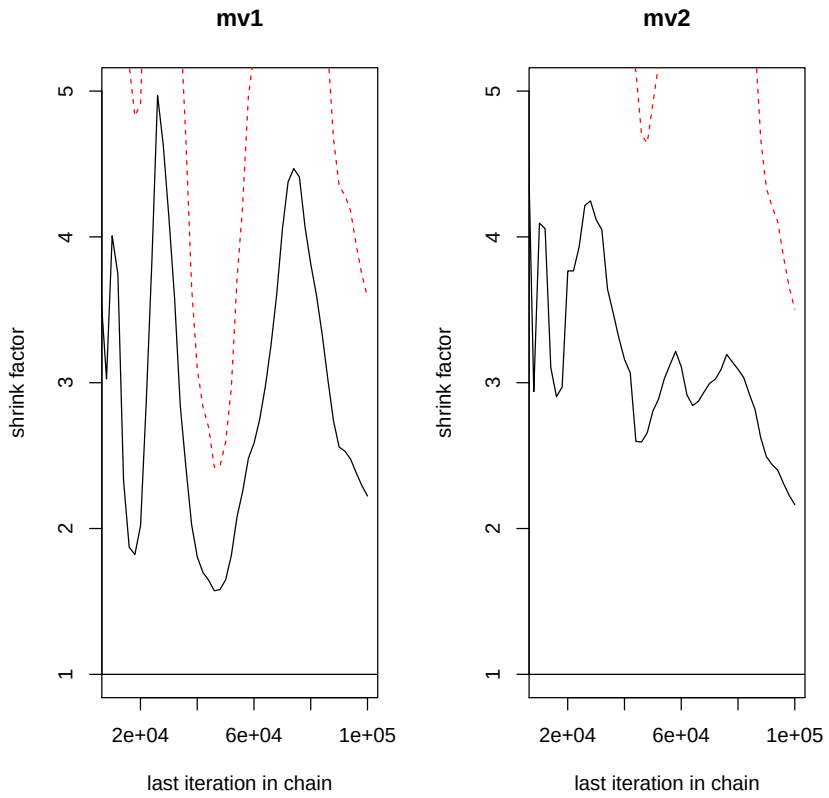


FIGURE 6.19 – Test de Gelman-Rubin pour des automates à 100 états sur un alphabet de 3 lettres, pour 100000 itérations, détails des résultats entre 10 000 et 100 000 itérations

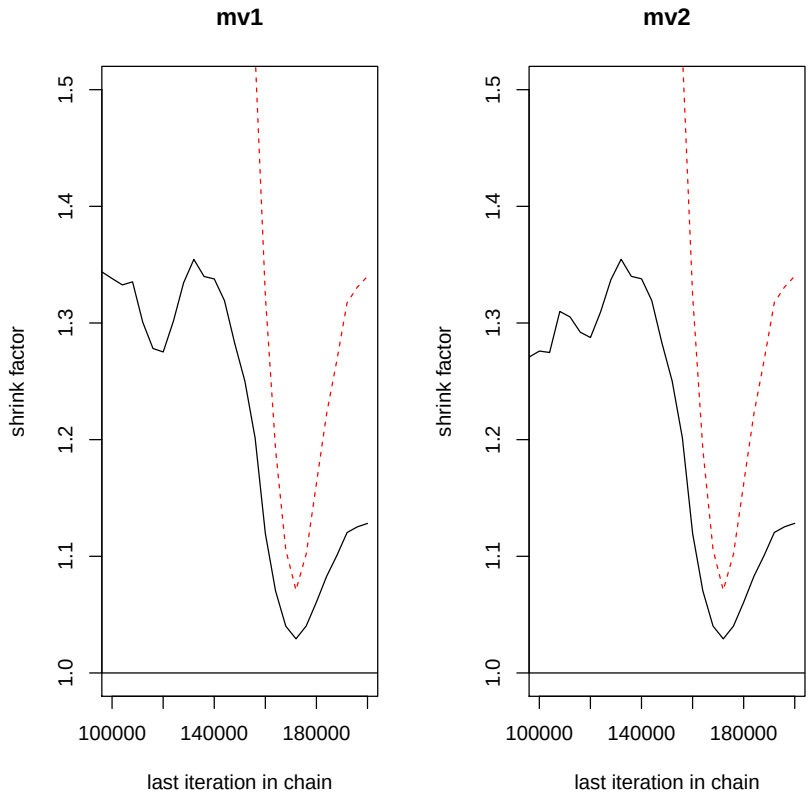


FIGURE 6.20 – Test de Gelman-Rubin pour des automates à 100 états sur un alphabet de 3 lettres, pour 200 000 itérations, détails des résultats entre 100 000 et 200 000 itérations

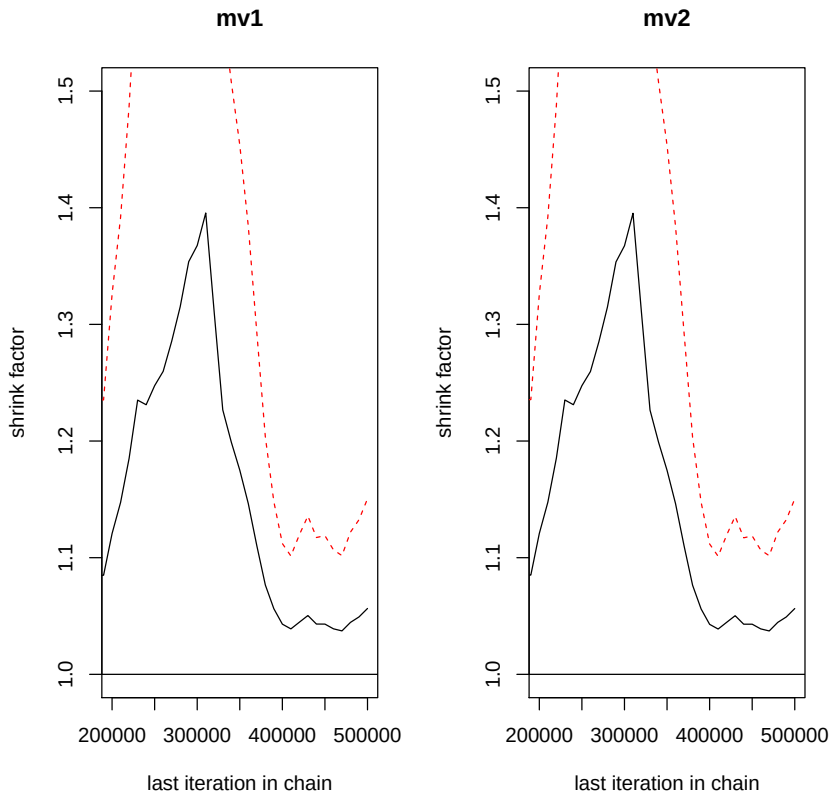


FIGURE 6.21 – Test de Gelman-Rubin pour des automates à 100 états sur un alphabet de 3 lettres, pour 500 000 itérations, détails des résultats entre 200 000 et 500 000 itérations

Conclusions et perspectives générales

La démarche présentée dans cette thèse consiste à mettre en perspective deux méthodes de générations aléatoires de structures combinatoires : la méthode récursive au sens large (dans le sens où les générateurs de Boltzmann en sont une extension) et la méthode de Monte-Carlo par Chaînes de Markov (MCMC).

Ces deux méthodes permettent, par des voies différentes, de construire des générateurs aléatoires d'objets appartenant à des classes pour lesquelles la combinatoire interdit une construction exhaustive de l'ensemble des objets et par là même l'utilisation d'un générateur pseudo aléatoire de nombres entiers. S'il fallait résumer grossièrement les choses, on pourrait dire que la stratégie de la méthode symbolique consiste à se *déplacer* dans l'arbre qui décrit la structure. La démarche par chaîne de Markov passe par la définition des modifications possibles des objets afin de construire une marche aléatoire dans le graphe des états. Avant toute autre analyse, le fait de disposer, dans les cas les plus favorables, de deux façons de construire un échantillonneur des objets d'une même classe, est déjà digne d'intérêt.

Un des buts de ce travail consistait à donner un prolongement aux travaux déjà cités à propos des graphes acycliques [CF11, CF12], pour approfondir ce que la génération MCMC pouvait apporter à la génération d'automates finis. L'utilisation de l'algorithme de Metropolis-Hastings pour tenir compte des classes d'isomorphismes confirme que la démarche semble pertinente.

Les conclusions vers lesquelles portent nos travaux consistent d'abord à dire que les méthodes par MCMC ne sont pas destinées à remplacer les outils très efficaces proposés par la méthode symbolique et en particulier les générateurs de Boltzmann. Ces outils ont sans doute atteint leur maturité et permettent une génération très efficace. Ils sont devenus *classiques* et doivent rester dans l'arsenal des approches à envisager lorsqu'il s'agit, pour le test, d'engendrer des classes a priori intéressantes d'automates, afin d'explorer des cas pratiques et de les analyser.

S'il ne faut pas oublier la méthode symbolique, il paraît néanmoins intéressant de retenir les techniques à base de la méthode de Monte-Carlo par chaîne de Markov pour l'échantillonnage de structures complexes. La méthode par chaîne de Markov introduit un nouvel angle d'attaque. Il ne s'agit plus de *démonter* la structure combinatoire de la classe d'objets que l'on souhaite échantillonner mais de définir les *mouvements possibles* dans la chaîne.

La souplesse de la méthode et sa relative facilité de mise en œuvre sont les deux qualités essentielles qui retiennent l'attention pour cette démarche. Il semble bien en effet qu'une fois envisagés les mouvements possibles dans le graphe associé à la chaîne, les hypothèses d'ergodicité ne soient pas trop délicates à démontrer ou à obtenir (en passant par exemple par la version *paresseuse* de la chaîne). Les deux écueils de la méthode ont déjà été soulignés :

- La difficulté à obtenir de façon théorique une majoration du temps de mélange.
- Le fait qu'il paraît clair, pour les générateurs de classes d'automates par exemple, d'envisager des temps de mélange meilleurs que $\mathcal{O}\left(n^2\right)$, où n est le nombre d'états

des automates tirés aléatoirement. Ce qui restreint l'outil à l'échantillonnage d'objets à au plus quelques milliers d'états, car il faut en général ajouter un temps de traitement au moins linéaire pour bouger dans la chaine.

Nous avons proposé dans ce travail une approche pour palier les inconvénients du premier écueil. La technique de *plongement* dans un produit d'hyper-cubes est sans doute une idée à retenir et à prolonger.

Si les méthodes de types MCMC sont destinées à se faire une place à coté des méthodes inspirées ou dérivées de la méthode symbolique, il faudra encore conquérir quelques résultats sur la majoration des temps de mélange. Nous avons déjà mentionné les techniques de couplage depuis le passé et de *strong stationary times* qui semblent prometteuses. Les techniques classiques de couplage posent des problèmes liés à la compatibilité des mouvements possibles depuis deux points de départ différents dans le graphe associé à la chaîne. Dès que les contraintes liées à la définition de la classe d'automate deviennent plus précises, le couplage classique devient une entreprise difficile qu'il ne faut sans doute pas abandonner *a priori*.

Un autre aspect de ce travail a consisté à essayer de mettre en valeur les outils statistiques accessibles pour appréhender les questions de temps de mélange dans les chaînes de Markov. Certes, ces outils ne donnent des réponses fiables que si celles-ci sont négatives. Dans ces cas, on dispose au moins de critère qui permettent d'abandonner rapidement une voie qui conduira à une branche morte. Dans le cas contraire, c'est-à-dire dans le cas où les résultats des tests d'auto-corrélation ou de Gelman-Rubin consistent à affirmer qu'*on ne peut pas dire que la chaîne mélange mal*, on sait au moins pouvoir raisonnablement poursuivre dans la voie dégagée.

Le chapitre 6 propose également, dans le cadre des tests statistiques, une démarche dans laquelle la méthode symbolique sert à construire le test du χ^2. Cette approche reste sans doute à approfondir. On peut en revanche d'ors et déjà insister sur le fait qu'un générateur uniforme d'objet d'une classe dont le cardinal est de l'ordre de 10^n avec n un entier de l'ordre de 10 à 100, ne peut avoir de sens que si on s'intéresse à une propriété de ces objets qui permettent de les ranger dans des classes d'équivalence dont on connaît la taille.

Sur ce point également, la démarche envisagée dans ce travail peut sans doute être ré-investie dans d'autres contextes.

Bibliographie

[Alb72] Albert R. Meyer and Larry J. Stockmeyer. The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Space. In *13th Annual Symposium on Switching and Automata Theory, College Park, Maryland, USA, October 25-27, 1972*, pages 125–129. IEEE Computer Society, 1972.

[Alt04] Altekar, Gautam and Dwarkadas, Sandhya and Huelsenbeck, John P. and Ronquist, Fredrik. Parallel Metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. *Bioinformatics*, 20(3) :407–415, 2004.

[AMR07] Marco Almeida, Nelma Moreira, and Rogério Reis. Enumeration and generation with a string automata representation. *Theor. Comput. Sci.*, 387(2) :93–102, 2007.

[And99] Andrieu, C. and Doucet, Arnaud. Joint Bayesian model selection and estimation of noisy sinusoids via reversible jump MCMC. *Signal Processing, IEEE Transactions on*, 47(10) :2667–2676, Oct 1999.

[And03] Andrieu, Christophe and de Freitas, Nando and Doucet, Arnaud and Jordan, MichaelI. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1-2) :5–43, 2003.

[Arf87] Mustapha Arfi. Polynomial Operations on Rational Languages. In Franz-Josef Brandenburg, Guy Vidal-Naquet, and Martin Wirsing, editors, *STACS 87, 4th Annual Symposium on Theoretical Aspects of Computer Science, Passau, Germany, February 19-21, 1987, Proceedings*, volume 247 of *Lecture Notes in Computer Science*, pages 198–206. Springer, 1987.

[BDN07] F. Bassino, J. David, and C. Nicaud. A Library to Randomly and Exhaustively Generate Automata. In *CIAA 2007*, volume 4783 of *Lecture Notes in Computer Science*, pages 303–305, 2007.

[BDN09] F. Bassino, J. David, and C. Nicaud. Enumeration and random generation of possibly incomplete deterministic automata. *Pure Mathematics and Applications*, 19 :1–16, 2009.

[BDN12] Frédérique Bassino, Julien David, and Cyril Nicaud. Average Case Analysis of Moore's State Minimization Algorithm. *Algorithmica*, 63(1-2) :509–531, 2012.

[BG98] Stephen P Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics*, 7(4) :434–455, 1998.

[BMT01] Ahmed Bouajjani, Anca Muscholl, and Tayssir Touili. Permutation rewriting and algorithmic verification. In *Logic in Computer Science, 2001. Proceedings. 16th Annual IEEE Symposium on*, pages 399–408. ieee, 2001.

[BN07] F. Bassino and C. Nicaud. Enumeration and random generation of accessible automata. *Theor. Comput. Sci.*, 381(1-3) :86–104, 2007.

[Boo78] Kellogg S. Booth. Isomorphism Testing for Graphs, Semigroups, and Finite Automata Are Polynomially Equivalent Problems. *SIAM J. Comput.*, 7(3) :273–279, 1978.

[Bré13] Pierre Brémaud. *Markov chains : Gibbs fields, Monte Carlo simulation, and queues*, volume 31. Springer Science & Business Media, 2013.

[CC96] Mary Kathryn Cowles and Bradley P. Carlin. Markov Chain Monte Carlo Convergence Diagnostics : A Comparative Review. *Journal of the American Statistical Association*, 91(434) :883, 1996.

[CF11] V. Carnino and S. De Felice. Random Generation of Deterministic Acyclic Automata Using Markov Chains. In *CIAA 2011*, volume 6807 of *Lecture Notes in Computer Science*, pages 65–75, 2011.

[CF12] Vincent Carnino and Sven De Felice. Sampling different kinds of acyclic automata using Markov chains. *Theor. Comput. Sci.*, 450 :31–42, 2012.

[CG95] S. Chib and E. Greenberg. Understanding the Metropolis-Hastings algorithm. *American Statistician*, 49 :327–335, 1995.

[CG09] Ben Calderhead and Mark Girolami. Estimating Bayes factors via thermodynamic integration and population MCMC . *Computational Statistics & Data Analysis* , 53(12) :4028 – 4045, 2009.

[CHM08] Gérard Cécé, Pierre-Cyrille Héam, and Yann Mainier. Efficiency of automata in semi-commutation verification techniques. *ITA*, 42(2) :197–215, 2008.

[CHPZ02] Jean-Marc Champarnaud, Georges Hansel, Thomas Paranthoën, and Djelloul Ziadi. Nfas bitstream-based random generation. In *Fourth International Workshop on Descriptional Complexity of Formal Systems - DCFS 2002*, pages 81–94, 2002.

[CN12] A. Carayol and C. Nicaud. Distribution of the number of accessible states in a random deterministic automaton. In *STACS 2012*, volume 14 of *LIPIcs*, pages 194–205, 2012.

[CP05] J.-M. Champarnaud and Th. Paranthoën. Random generation of DFAs. *Theor. Comput. Sci.*, 330(2) :221–235, 2005.

[CTV15] Julien Clément, TH Nguyen Thi, and Brigitte Vallée. Towards a realistic analysis of some popular sorting algorithms. *Combinatorics, Probability and Computing*, 24(01) :104–144, 2015.

[Dav10] Julien David. *Génération aléatoire d'automates et analyse d'algorithmes de minimisation.* PhD thesis, Université Paris-Est de Marne-la-Vallée, 2010.

[DBL13] *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1 : Long Papers.* The Association for Computer Linguistics, 2013.

[DFLS04] Ph. Duchon, Ph. Flajolet, G. Louchard, and G. Schaeffer. Boltzmann Samplers for the Random Generation of Combinatorial Structures. *Combinatorics, Probability & Computing*, 13(4-5) :577–625, 2004.

[DFN13] Sven De Felice and Cyril Nicaud. Random Generation of Deterministic Acyclic Automata Using the Recursive Method. In *Computer Science Theory and Applications*, volume 7913 of *Lecture Notes in Computer Science*, pages 88–99. 2013.

[DZ99] Alain Denise and Paul Zimmermann. Uniform random generation of decomposable structures using floating-point arithmetic. *Theoretical Computer Science* , 218(2) :233 – 248, 1999.

[Fla90] Flajolet, Philippe and Zimmermann, Paul and Salvy, Bruno. Automatic average-case analysis of algorithms. Research Report RR-1233, INRIA, 1990. Projet ICSLA.

[FPSV09] Pasquale Foggia, Gennaro Percannella, Carlo Sansone, and Mario Vento. Benchmarking graph-based clustering algorithms. *Image Vision Comput.*, 27(7) :979–988, 2009.

[FS08] Ph. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2008.

[FWW97] A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems (extended abstract). In *INFINITY'97)*, volume 9 of *Electronic Notes in Theoretical Computer Science*, pages 27–39, 1997.

[Gal14] Joseph A. Gallian. A Dynamic Survey of Graph Labeling. *The Electronic Journal of Combinatorics*, 17, 2014.

[Gib02] Gibbs, Alison L. and Su, Francis Edward. On Choosing and Bounding Probability Metrics. *International Statistical Review*, 70(3) :419–435, 2002.

[HÓ3] Pierre-Cyrille Héam. Some complexity results for polynomial rational expressions. *Theor. Comput. Sci.*, 1-3(299) :735–741, 2003.

[HNS10] P.-C. Héam, C. Nicaud, and S. Schmitz. Parametric random generation of deterministic tree automata. *Theor. Comput. Sci.*, 411(38-39) :3469–3480, 2010.

[Jor02] Joris van der Hoeven. Relax, but Don't be Too Lazy . *Journal of Symbolic Computation* , 34(6) :479 – 542, 2002.

[Jul12] Julien David. Average complexity of Moore's and Hopcroft's algorithms. *Theor. Comput. Sci.*, 417 :50–65, 2012.

[Kor78] D. Korshunov. Enumeration of Finite Automata. *Problemy Kibernetiki*, 34 :5–82, 1978.

[Kor86] A. D. Korshunov. On the Number of Non-isomorphic Strongly Connected Finite Automata. *Elektronische Informationsverarbeitung und Kybernetik*, 22(9) :459–462, 1986.

[Leb14] Lebeau, Gilles and Michel, Laurent. hypoelliptic random walks. *Journal de l' Institut de Mathématiques de Jussieu*, pages 1–41, 2014.

[Lej04] Lejeune, M. *Statistique : la théorie et ses applications*. Collection Statistique et probabilités appliquées. Springer, 2004.

[LPW09] David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov chains and mixing times*. American Mathematical Soc., 2009.

[Luk82] Eugene M. Luks. Isomorphism of Graphs of Bounded Valence can be Tested in Polynomial Time. *J. Comput. Syst. Sci.*, 25(1) :42–65, 1982.

[Mat79] Rudolf Mathon. A Note on the Graph Isomorphism counting Problem. *Inf. Process. Lett.*, 8(3) :131–132, 1979.

[Nic00] C. Nicaud. *Étude du comportement en moyenne des automate finis et des langages rationnels*. PhD thesis, 2000.

[Nic08] Nichols, Johanna and Warnow, Tandy. Tutorial on Computational Linguistic Phylogeny. *Language and Linguistics Compass*, 2(5) :760–820, 2008.

[Nic14] Cyril Nicaud. Random Deterministic Automata. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 Symposium, MFCS 2014*, volume 8634 of *Lecture Notes in Computer Science*, pages 5–23. Springer, 2014.

[NPR10] Cyril Nicaud, Carine Pivoteau, and Benoît Razet. Average Analysis of Glushkov Automata under a BST-Like Model. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010*, LIPIcs, pages 388–399, 2010.

[Par10] Parosh Aziz Abdulla and Yu-Fang Chen and Lukás Holík and Richard Mayr and Tomás Vojnar. When Simulation Meets Antichains. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS*

2010, Paphos, Cyprus, March 20-28, 2010. Proceedings, volume 6015 of *Lecture Notes in Computer Science*, pages 158–174. Springer, 2010.

[Phi94] Philippe Flajolet and Paul Zimmermann and Bernard Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science* , 132(1–2) :1 – 35, 1994.

[Piv08] Carine Pivoteau. *Génération aléatoire de structures combinatoires : méthode de Boltzmann effective*. PhD thesis, Université Pierre et Marie Curie - Paris VI, 2008.

[R D08] R Development Core Team. *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[Ram00] Ramesh R. Sarukkai. Link prediction and path analysis using Markov chains1 . *Computer Networks* , 33(1–6) :377 – 386, 2000.

[Riv12] Rivoirard, V. and Stoltz, G. *Statistique mathématique en action : cours, problèmes d'application corrigés et mises en action concrètes*. Mathématiques concrètes. Vuibert, 2012.

[Sai96] Saito, Yoshihiro and Mitsui, Taketomo. Stability analysis of numerical schemes for stochastic differential equations. *SIAM Journal on Numerical Analysis*, 33(6) :2254–2267, 1996.

[Sak09] J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[STV01] Thomas Schwentick, Denis Thérien, and Heribert Vollmer. Partially-Ordered Two-Way Automata : A New Characterization of DA. In *Developments in Language Theory, 5th International Conference, DLT 2001, Vienna, Austria, July 16-21, 2001, Revised Papers*, volume 2295 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2001.

[TV05] Deian Tabakov and Moshe Y. Vardi. Experimental Evaluation of Classical Automata Constructions. In Geoff Sutcliffe and Andrei Voronkov, editors, *LPAR'05,*, volume 3835 of *Lecture Notes in Computer Science*, pages 396–411. Springer, 2005.

Table des figures

Liste des tableaux

Document réalisé avec LaTeX et :
le style LaTeX pour Thèse de Doctorat créé par S. Galland — http://www.multiagent.fr/ThesisStyle
la collection de paquets `tex-upmethodology` — http://www.arakhne.org/tex-upmethodology/

Résumé :

Le concept d'automate, central en théorie des langages, est l'outil d'appréhension naturel et efficace de nombreux problèmes concrets. L'usage intensif des automates finis dans un cadre algorithmique s'illustre par de nombreux travaux de recherche. La correction et l'évaluation sont les deux questions fondamentales de l'algorithmique. Une méthode classique d'évaluation s'appuie sur la génération aléatoire contrôlée d'instances d'entrée.Les travaux décrits dans cette thèse s'inscrivent dans ce cadre et plus particulièrement dans le domaine de la génération aléatoire uniforme d'automates finis.

L'exposé qui suit propose d'abord la construction d'un générateur aléatoire d'automates à pile déterministes, *real time*. Cette construction s'appuie sur la méthode symbolique. Des résultats théoriques et une étude expérimentale sont exposés.

Un générateur aléatoire d'automates non-déterministes illustre ensuite la souplesse d'utilisation de la méthode de Monte-Carlo par Chaînes de Markov (MCMC) ainsi que la mise en œuvre de l'algorithme de Metropolis-Hastings pour l'échantillonnage à isomorphisme près. Un résultat sur le temps de mélange est donné dans le cadre général.

L'échantillonnage par méthode MCMC pose le problème de l'évaluation du temps de mélange dans la chaîne. En s'inspirant de travaux antérieurs pour construire un générateur d'automates partiellement ordonnés, on montre comment différents outils statistiques permettent de s'attaquer à ce problème.

Abstract:

This is the abstract in English
The concept of automata, central to language theory, is the natural and efficient tool to apprehend various practical problems.
The intensive use of finite automata in an algorithmic framework is illustrated by numerous research works.
The correctness and the evaluation of performance are the two fundamental issues of algorithmics.
A classic method to evaluate an algorithm is based on the controlled random generation of inputs.
The work described in this thesis lies within this context and more specifically in the field of the uniform random generation of finite automata.

The following presentation first proposes to design a deterministic, real time, pushdown automata generator. This design builds on the symbolic method. Theoretical results and an experimental study are given.

This design builds on the symbolic method. Theoretical results and an experimental study are given. A random generator of non deterministic automata then illustrates the flexibility of the Markov Chain Monte Carlo methods (MCMC) as well as the implementation of the Metropolis-Hastings algorithm to sample up to isomorphism. A result about the mixing time in the general framework is given.

The MCMC sampling methods raise the problem of the mixing time in the chain. By drawing on works already completed to design a random generator of partially ordered automata, this work shows how various statistical tools can form a basis to address this issue.

■ École doctorale SPIM 16 route de Gray F - 25030 Besançon cedex
■ tél. +33 (0)3 81 66 66 02 ■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr