UFR de Sciences **Université de Caen Basse–Normandie**
École doctorale SIMEM

Thèse présentée par
M. Julien CLÉMENT
et soutenue le 22 septembre 2000
en vue de l'obtention du
DOCTORAT de l'UNIVERSITÉ de CAEN
Spécialité Informatique
(Arrêté du 30 mars 1992)

Arbres Digitaux et Sources Dynamiques

Composition du jury

Rapporteurs :	Maxime CROCHEMORE, Professeur à l'Université de Marne-la-Vallée Robert SEDGEWICK, Professeur à l'université de Princeton (USA) Wojciech SZPANKOWSKI, Professeur à l'Université de Purdue (USA)
Examinateurs :	Maxime CROCHEMORE, Professeur, Université de Marne-la-Vallée Philippe FLAJOLET, Directeur de Recherche, INRIA Rocquencourt (co-directeur) Christiane FROUGNY, Professeur, Université de Paris VIII Etienne GRANDJEAN, Professeur, Université de Caen Bernard PRUM, Professeur, Université d'Évry Val d'Essonne Wojciech SZPANKOWSKI, Professeur, Université de Purdue (USA) Brigitte VALLÉE, Professeur, Université de Caen (directeur) Paul ZIMMERMANN, Directeur de recherche, INRIA Lorraine

Groupe de Recherche en Informatique, Image, Instrumentation de Caen
CNRS UPRESA 6072

UFR de Sciences **Université de Caen Basse–Normandie**
École doctorale SIMEM

Thèse présentée par
M. Julien CLÉMENT
et soutenue le 22 septembre 2000
en vue de l'obtention du
DOCTORAT de l'UNIVERSITÉ de CAEN
Spécialité Informatique
(Arrêté du 30 mars 1992)

Arbres Digitaux et Sources Dynamiques

Composition du jury

Rapporteurs :	Maxime CROCHEMORE, Professeur à l'Université de Marne-la-Vallée Robert SEDGEWICK, Professeur à l'université de Princeton (USA) Wojciech SZPANKOWSKI, Professeur à l'Université de Purdue (USA)
Examinateurs :	Maxime CROCHEMORE, Professeur, Université de Marne-la-Vallée Philippe FLAJOLET, Directeur de Recherche, INRIA Rocquencourt (co-directeur) Christiane FROUGNY, Professeur, Université de Paris VIII Etienne GRANDJEAN, Professeur, Université de Caen Bernard PRUM, Professeur, Université d'Évry Val d'Essonne Wojciech SZPANKOWSKI, Professeur, Université de Purdue (USA) Brigitte VALLÉE, Professeur, Université de Caen (directeur) Paul ZIMMERMANN, Directeur de recherche, INRIA Lorraine

Groupe de Recherche en Informatique, Image, Instrumentation de Caen
CNRS UPRESA 6072

Remerciements

Je souhaite en premier lieu remercier Brigitte Vallée qui à travers ses enseignements a su m'attirer vers l'algorithmique en général et l'analyse en moyenne en particulier. J'ai été ravi de l'avoir comme directrice de thèse et n'ai jamais été déçu par ses qualités humaines et scientifiques. Je louerai longtemps son optimisme à toute épreuve et son énergie communicative.

Je suis très reconnaissant à Philippe Flajolet de m'avoir accueilli au projet ALGO et d'avoir accepté d'être co-directeur de ma thèse. Le côtoyer a été réellement une expérience passionnante. Sa connaissance «monstrueuse» et approfondie de nombreux domaines a été très enrichissante pour moi. Je pense que tous ceux qui ont eu l'occasion d'avoir une conversation avec lui me comprendront. Son extrême gentillesse et sa compréhension m'ont également été très précieuses.

Maxime Crochemore, Robert Sedgewick et Wojciech Szpankowski m'ont fait le grand honneur d'être rapporteurs de cette thèse. Je les remercie très vivement pour le temps qu'ils m'ont accordé : leurs commentaires me furent fort utiles.

Ma reconnaissance va également à Christiane Frougny, Etienne Grandjean, Bernard Prum et Paul Zimmermann pour avoir accepté d'être membres de mon jury. Je suis flatté de l'intérêt qu'ils portent à mon travail.

Je remercie tous les membres du GREYC pour m'avoir accueilli au sein du laboratoire. Plus personnellement, je tiens à remercier Ali, Jérémie, Samuel, Samuel, Séverine, Arnaud, Jean-Marie et Véronique.

Je ne saurais oublier les personnels administratifs du laboratoire pour leur aide et leur gentillesse : Sylvie, Séverine, Françoise, Hélène et Chantal. J'adresse également mes plus sincères remerciements à Virginie ainsi qu'à tous les membres du projet ALGO.

Table des matières

Chapitre 1

Introduction

Motivations. Les arbres digitaux, encore appelés tries, sont une structure de données centrale en informatique. Ils reproduisent un principe utilisé par exemple dans les dictionnaires ou les répertoires téléphoniques. Les noms sont regroupés selon leur première lettre, ce qui permet un accès rapide (grâce à un onglet par exemple). Ce mécanisme peut être poursuivi récursivement jusqu'à séparer les mots les uns des autres : on aboutit à une structure de trie. Étant donné un alphabet fixé $\mathcal{M} = \{a_1, \ldots, a_r\}$, et en notant Y un ensemble de mots infinis sur l'alphabet $\mathcal{M}$, le trie associé à Y est défini récursivement par la règle,

$$\mathsf{trie}(Y) = \langle \mathsf{trie}(Y \setminus a_1), \ldots, \mathsf{trie}(Y \setminus a_r) \rangle,$$

où $Y \setminus \alpha$ signifie le sous-ensemble de Y contenant les mots qui commencent par α et dont on a retiré le premier symbole α. La récursion est arrêtée dès lors que Y contient moins de deux éléments. L'avantage du trie est qu'il ne considère que l'ensemble minimal des préfixes nécessaires pour distinguer l'ensemble des éléments de Y.
La recherche d'un mot consiste en l'examen successif de ses lettres. Chaque lettre supplémentaire conduit à restreindre l'ensemble des mots qui partagent le même préfixe. Dans l'arbre, cela correspond à cheminer le long d'une branche. Les opérations d'insertion et de suppression d'un mot peuvent être également facilement implantées. La structure de trie est donc particulièrement adaptée pour les applications de type dictionnaire, y compris dans un contexte dynamique où l'ensemble des mots n'est pas «figé» mais peut varier énormément par insertions et suppressions successives de mots.

Dans cette thèse, nous envisageons ces structures dans le cadre de l'analyse en moyenne. À partir d'un modèle aléatoire sur les entrées, on étudie les comportements fins de la structure. Les tries ont beaucoup été étudiés (et le sont encore), mais dans la grande majorité des cas en considérant un modèle probabiliste simple. L'ensemble des mots est produit à l'aide d'une source classique (source sans mémoire où la probabilité d'apparition d'un symbole est indépendante du passé ou encore les chaînes de Markov qui prennent en compte un passé fini). Cela désigne le cadre classique d'analyse en moyenne des tries.
Les considérations pratiques liées à ces structures sont immédiates :
– l'étude en moyenne est réalisée dans un modèle classique. Peut-on considérer d'autres types de sources, si possible plus générales ou réalistes ?
– Comment implanter en machine une structure de trie abstraite ?
La deuxième question a été partiellement traitée par Bentley et Sedgewick [8]. Ces deux auteurs, suivant une idée de Clampett [15] introduisent la structure de TST (*ternary search trie*) qui considère un principe ternaire de branchement afin de représenter efficacement un trie. L'expérimentation sur des données réelles montre qu'il s'agit d'une structure de données parmi les plus

efficaces pour traiter des données textuelles. Les auteurs concluent en fait qu'on peut avantageusement utiliser les TST en lieu et place des tables de hachage dans bon nombre d'applications. Cette démarche expérimentale (appuyée par une étude dans le pire des cas) donne une intuition quant au bon comportement de la structure mais ne constitue pas une preuve.

À la même époque, je m'intéressais aux mêmes questions, aiguillé en cela par l'étude en moyenne des tries de nombres basés sur la numération en fraction continue. En effet, la source considérée (le développement en fraction continue) est l'exemple typique d'une source de symboles ne rentrant pas dans le cadre classique. En particulier l'alphabet est l'ensemble des entiers naturels (donc non borné) et les dépendances entre symboles sont également non bornées (contrairement au cas markovien). Les questions pratiques liées à l'implantation de la structure d'arbre se posent naturellement.
L'exemple de la fraction continue illustre la démarche de cette thèse. Nous proposons deux axes de généralisation :

– Le premier axe concerne le modèle aléatoire. À la question «comment produire les mots ?», on s'inspire du domaine des systèmes dynamiques en physique statistique. Un mot n'est autre que la succession d'états dans un système dynamique. Il s'agit ici d'une généralisation, puisque ce modèle de source dit «dynamique» englobe les sources sans mémoire, les chaînes de Markov, et également la source en fraction continue du paragraphe précédent. De plus, ce modèle permet également de considérer une certaine densité pour la configuration initiale du système dynamique. On obtient alors une source dynamique *probabilisée*.
– Le deuxième axe concerne la structure elle-même. De quelle façon la structure de trie, abstraite, peut-elle être effectivement représentée en machine ? De façon un peu inattendue, une solution élégante à cette question (celle-là même retenue par Bentley et Sedgewick pour construire les TST) consiste à mettre en vis-à-vis deux structures que tout oppose conceptuellement : les arbres binaires de recherche et les tries. Ces structures d'arbre, figurant parmi les plus étudiées et les plus utilisées, retiennent chacune la quintessence de deux principes de construction bien distincts. L'arbre binaire de recherche s'appuie sur l'ordre relatif à l'intérieur de l'univers des clés (les éléments à stocker dans l'arbre) tandis que le trie considère les clés au travers de leurs représentations. Ces deux structures se rejoignent dans cette thèse. Les tries restent le principal objet d'étude, mais les arbres binaires de recherche apparaissent sous un jour nouveau. Le rapprochement des deux structures s'avère «algorithmiquement» intéressant. La structure de *ternary search trie*, hybridation d'une structure globale de trie et de multiples arbres binaires de recherche locaux, fournit une structure de données efficace. Les TST s'avèrent en pratique au moins aussi efficaces que les tables de hachage tout en offrant des fonctionnalités supplémentaires.
  Nous envisagerons d'autres hybridations comme celle résultant d'une structure de trie et d'une liste chaînée ordonnée, ou encore de la représentation classique des tries grâce à des tableaux de pointeurs. Ce sont donc réellement des tries généraux hybrides qui sont considérés dans cette thèse, afin de se rapprocher d'une implantation réelle de la structure de trie.

C'est du point de vue de l'analyse en moyenne que nous étudions ces deux généralisations. Le modèle de source, très général et flexible, permet à la fois de proposer un modèle proche de la réalité et un cadre d'analyse élégant. Nous nous attachons essentiellement à comparer différents types de représentation (les hybridations les plus naturelles du trie) avec ce modèle de source, ce qui nous permettra de confirmer par l'analyse la suprématie de la structure de TST.

Méthodes. Nous utilisons des méthodes variées allant des méthodes probabilistes aux méthodes analytiques avec une excursion originale vers l'analyse fonctionnelle.
Du fait des généralisations proposées pour la source et l'hybridation des tries, nous introduisons des outils jusque là non employés en analyse en moyenne. Nous commençons par rappeler le schéma classique d'analyse en moyenne pour les tries afin de mieux situer où interviennent les

modifications dues à l'introduction d'un nouveau type de source et l'hybridation.

L'analyse débute par le calcul d'une expression exacte de la valeur moyenne de divers paramètres (la hauteur, le nombre de nœuds internes ou encore la longueur de cheminement). Les méthodes employées peuvent être probabilistes ou encore faire appel aux techniques de séries génératrices. Nous obtenons des expressions où interviennent les probabilités qu'un mot commence par un préfixe donné, ce que nous appelons les *probabilités d'apparition de préfixes*. Si les résultats obtenus sont exacts, ils restent peu informatifs dans le sens où il est difficile de voir directement les ordres de grandeur impliqués. Ceci justifie l'étude du comportement asymptotique. Les résultats précédents, et cela est caractéristique des analyses autour de structures digitales, s'expriment à l'aide de sommes dites «harmoniques». Une somme harmonique est une somme de la forme

$$F(x) = \sum_{k \in K} \lambda_k f(\mu_k x), \tag{1.1}$$

où les familles $(\lambda_k)_{k\in K}$ et $(\mu_k)_{k\in K}$ représentent respectivement les «amplitudes» et les «fréquences», et f est appelée «fonction de base». Ici les amplitudes comme les fréquences s'expriment en fonction des probabilités d'apparition de préfixes. Un outil classique pour l'analyse asymptotique de sommes harmoniques est la transformée de Mellin. Elle permet en effet de passer (dans tous les modèles de source) d'une forme close du type de (1.1) à une formule asymptotique en passant par l'étude des pôles de la *série de Dirichlet* associée

$$\Theta(s) = \sum_{k \in K} \lambda_k \mu_k^s.$$

C'est à ce niveau que les deux cadres d'analyse divergent. Dans le cadre classique, les probabilités d'apparition de préfixes et les séries de Dirichlet associées ont une expression simple et explicite qui permet de conclure l'analyse grâce des techniques d'analyse élémentaires. Souvent les résultats obtenus le sont dans un modèle aléatoire dit de Poisson, un modèle simplifiant l'analyse mais qui rend les résultats moins «lisibles». Une dernière étape de *dépoissonisation* permet d'aboutir au résultat final.

Le schéma non classique des sources dynamiques probabilisées et des tries hybrides ajoute au schéma précédent des techniques d'analyse fonctionnelle. La différence majeure avec le cas classique survient lors de l'analyse asymptotique. Les probabilités d'apparition de préfixes et donc les séries de Dirichlet mises en jeu n'admettent plus d'expressions simples. Dans le contexte des sources dynamiques, la probabilité d'apparition d'un préfixe correspond exactement à la mesure d'un intervalle appelé *intervalle fondamental* (l'intervalle est donc l'ensemble des états initiaux du système qui passent par les états successifs correspondant aux lettres du préfixe). Nous appellerons encore cette probabilité la *mesure fondamentale* pour ce préfixe. Nous avons donc besoin d'engendrer les mesures fondamentales puisque ce sont celles-ci qui interviennent dans les séries de Dirichlet. Nous introduisons à cette fin des *opérateurs générateurs*. Ces opérateurs introduisent des méthodes d'analyse fonctionnelle et sont directement inspirés des opérateurs de transfert de Ruelle. Historiquement, ces opérateurs dérivent des opérateurs transformateurs de densité. Ruelle, dans le cadre des systèmes dynamiques, a étendu ces opérateurs en des opérateurs de transfert en leur adjoignant un paramètre supplémentaire lié à la température du système. Brigitte Vallée a initié l'utilisation de tels opérateurs dans le cadre de l'analyse d'algorithme.
Nous introduisons dans cette thèse une nouvelle extension des opérateurs de transfert de Ruelle adaptée aux tries hybrides. On obtient des opérateurs multisécants opérant sur des espaces de fonctions de plusieurs variables. Ces nouveaux opérateurs étendent de façon appropriée les opérateurs de transfert, tout en gardant de bonnes propriétés spectrales.
Ces opérateurs engendrent les mesures fondamentales, ce qui donne une expression alternative des séries de Dirichlet associées au problème. Nous pouvons en utilisant les propriétés spectrales

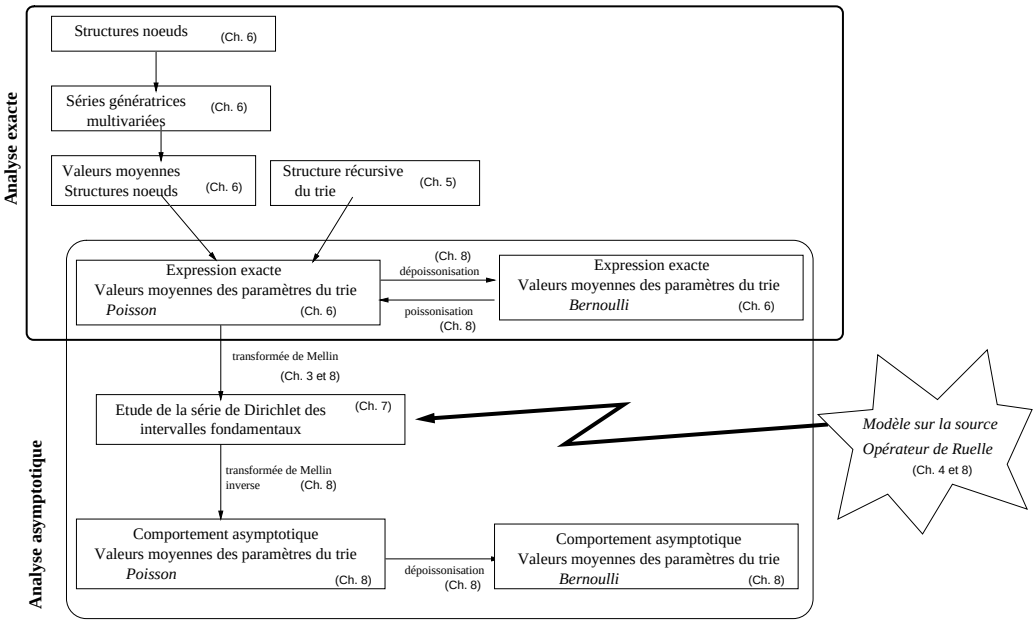


FIG. 1.1 – Les différentes étapes de l'analyse.

dominantes localiser assez précisément les pôles de ces séries. Ainsi malgré la généralité de la source, nous obtenons grâce une analyse de Mellin légèrement adaptée des résultats finaux précis. Là encore, une dernière étape de dépoissonisation est nécessaire pour passer du modèle de Poisson vers le modèle plus «naturel» dit de Bernoulli. Outre des méthodes classiques comme la méthode du point de col, nous introduisons la dépoissonisation de Dirichlet.

Résultats. Les résultats sont multiples et se situent à plusieurs niveaux. D'abord, et c'est d'ailleurs l'intérêt du modèle de source considéré, l'existence de propriétés spectrales dominantes permet d'exprimer les résultats à l'aide de constantes caractéristiques intrinsèques de la source. Nous exprimons ces constantes grâce à l'un des objets spectraux dominants omniprésent dans cette thèse, la fonction valeur propre $\lambda(s)$ (définie sur un voisinage de l'axe réel). Les constantes associées à une source S, l'entropie $h(\mathsf{S})$ et la probabilité de coïncidence $c(\mathsf{S})$, sont reliées à cette fonction par les formules

$$h(\mathsf{S}) = -\lambda'(1), \qquad c(\mathsf{S}) = \lambda(2).$$

Les résultats principaux concernent un trie aléatoire construit sur n mots produits par une source dynamique probabilisée.

(i) La hauteur d'un trie standard a une espérance en $\log n$ et la loi de distribution est de type double exponentielle,

$$\mathrm{E}[h_n] \sim \frac{2}{|\log c(\mathsf{S})|}\,\log n$$
$$\lim_{n\to\infty} \sup_{k\geq 0} \left|\Pr\{h_n \leq k\} - \exp\left[-\rho c(\mathsf{S})^k n^2\right]\right| = 0,$$

où ρ est une constante positive relative à la source S et à la densité initiale.

(ii) La valeur moyenne du nombre de nœuds internes du trie (appelé également la taille du trie) est, à quelques fluctuations près, linéaire par rapport à n le nombre de mots stockés

$$S(n) \approx \frac{1}{h(\mathsf{S})}\, n.$$

(iii) La valeur moyenne de la longueur de cheminement dépend du type d'hybridation choisi pour le trie. Désignons par $\langle A\rangle$ l'hybridation utilisant des tableaux de pointeurs, par $\langle L\rangle$ celle qui utilise des listes ordonnées et par $\langle B\rangle$ celle qui utilise les arbres binaires de recherche. Pour toute source définie à partir d'un alphabet fini, les valeurs moyennes des longueurs de cheminement sont toutes en $n \log n$,

$$P_A(n) \sim \frac{1}{h(\mathsf{S})} n \log n, \qquad P_L(n) \sim \frac{K_L(\mathsf{S})}{h(\mathsf{S})} n \log n, \qquad P_B(n) \sim \frac{K_B(\mathsf{S})}{h(\mathsf{S})} n \log n,$$

avec des constantes explicites $K_L(\mathsf{S}), K_B(\mathsf{S})$ dépendant uniquement du mécanisme de la source S (et pas de la densité initiale). Les résultats sont à partager en deux catégories : les paramètres additifs regroupant les paramètres de taille et de longueur de cheminement, et le paramètre de la hauteur.
Dans le cas d'un alphabet infini, le trie représenté à l'aide tableaux de pointeurs ne correspond plus à rien en tant que structure de données. Dans ce cas, les ordres de grandeur des valeurs moyennes des longueurs de cheminement des tries en liste et des tries en ABR peuvent être différents. Le mécanisme de source basé sur les fractions continues constitue d'ailleurs un exemple d'un tel phénomène.

Nous avons mentionné ci-dessus les résultats centraux de cette thèse. Par ailleurs, la chaîne d'analyse suivie fournit une série d'autres résultats qui ont leur intérêt propre et que nous ne détaillerons pas dans cette introduction (ceux-ci concernent les diverses combinaisons entre modèles aléatoires, analyse exacte ou asymptotique et les paramètres de tries).

De plus, l'analyse fait une incursion inédite dans les arbres binaires de recherche. Pour le modèle, très largement utilisé en pratique, des arbres binaires de recherche où l'égalité des clés est permise, nous fournissons de nouveaux résultats grâce à une méthode symbolique originale.

Enfin, une étude expérimentale à partir du livre d'H. Melville *Mobydick* corrobore qualitativement l'analyse en moyenne. Cette étude valide également l'idée consistant à hybrider les tries avec des arbres binaires de recherche. En pratique, cela s'avère un élégant compromis entre efficacité en temps et en espace. Le correcteur orthographique `epelle`, réalisé en collaboration avec P. Zimmermann [17], constitue un autre exemple pratique où la structure de TST est mise à profit (en utilisant de plus une forme compressée).

Les résultats de cette thèse ont fait l'objet de deux parutions. Le premier [16] traite des tries hybrides avec un modèle de source classique (source sans mémoire et chaîne de Markov). Le second article [18] reprend ces résultats en les élargissant aux sources dynamiques.

Chapitre 2

La structure de trie en algorithmique

Sommaire

De nombreuses structures d'arbre existent en informatique. On peut grossièrement séparer ces arbres en deux familles. Dans la première, les éléments stockés dans l'arbre sont considérés de façon «relative». Les éléments sont des atomes (dans le sens informaticien du terme, c.-à-d. indivisibles) et les opérations sur les éléments comme la comparaison, l'échange ou l'affectation sont également considérées comme des opérations atomiques. Ces structures se fondent sur des comparaisons sur les éléments de l'univers des clés pour retrouver une information et font donc partie des *structures de données fondées sur des comparaisons* [76]. L'autre famille d'arbres rassemble les structures arborescentes où l'on profite au contraire d'une décomposabilité des éléments (on peut décomposer un mot en lettres, un vecteur en ces composantes) et se rapproche donc des structures de données axées sur la représentation.

Les arbres binaires de recherche ou les tas (utilisés pour implanter les files de priorité) constituent des exemples d'arbres appartenant à la première catégorie. Les arbres digitaux[1], et les tries, sont l'archétype de la deuxième famille d'arbres.

Nous décrivons dans un premier temps les différentes variétés et/ou extensions des arbres digitaux. Dans un deuxième temps, nous détaillerons quelques applications.

2.1 Arbres digitaux et tries

Le terme «trie» provient de la contraction de deux mots anglais : «tree» et «retrieval» et a été introduit par Fredkin [47]. C'est un arbre planaire (c.-à.-d dont les fils sont ordonnés), qui est utilisé dans la recherche lexicographique ou plus généralement pour les applications de type dictionnaire. La structure de trie est une structure de donnée dynamique adaptée pour représenter *un ensemble de mots* qui évolue, car il est facile de supprimer, rechercher, ou encore ajouter un mot (voir la section 2.3 sur les applications).

2.1.1 Définitions

Soit un alphabet $\mathcal{M}$ inclus dans $\mathbb{N}$ fini ou dénombrable, de cardinal $r \in \mathbb{N}^* \cup \{+\infty\}$. L'ensemble des mots infinis $\mathcal{M}^{\mathbb{N}}$ est défini par

$$\mathcal{M}^{\mathbb{N}} = \{\boldsymbol{m} = m_1 m_2 \dots \mid \forall j \geq 1,\ m_j \in \mathcal{M}\}.$$

Pour un mot $\boldsymbol{w}$ infini, on peut définir la notion de *préfixe*. Un mot $\boldsymbol{v}$ est un préfixe d'un mot $\boldsymbol{w}$ s'il existe un mot $\boldsymbol{u}$ tel que l'on ait $\boldsymbol{w} = \boldsymbol{v}\cdot\boldsymbol{u}$. Le mot $\boldsymbol{u}$ dans cette décomposition est appelé *suffixe*.

On définit les fonctions *tête* $\underline{\sigma} : \mathcal{M}^{\mathbb{N}} \to \mathcal{M}$ et *queue* $\underline{\mathrm{T}} : \mathcal{M}^{\mathbb{N}} \to \mathcal{M}^{\mathbb{N}}$ (qu'on aurait aussi bien pu nommer `car` et `cdr` en référence au langage LISP) par

$$\begin{aligned} \underline{\sigma} :&\quad \boldsymbol{m} = m_1 m_2 \cdots \mapsto \underline{\sigma}(\boldsymbol{m}) = m_1, \\ \underline{\mathrm{T}} :&\quad \boldsymbol{m} = m_1 m_2 \cdots \mapsto \underline{\mathrm{T}}(\boldsymbol{m}) = m_2 m_3 \cdots . \end{aligned}$$

Remarque – L'application $\underline{\mathrm{T}}$ définit une opération de décalage.

On introduit également la notation $\underline{\mathrm{T}}_{[\alpha]}$ pour $\alpha \in \mathcal{M}$ afin de désigner la restriction de la fonction $\underline{\mathrm{T}}$ aux mots qui commencent par le symbole α. Pour $\alpha \in \mathcal{M}$, on a l'application $\underline{\mathrm{T}}_{[\alpha]} : \alpha\cdot\mathcal{M}^{\mathbb{N}} \to \mathcal{M}^{\mathbb{N}}$ définie par

$$\underline{\mathrm{T}}_{[\alpha]} :\quad \boldsymbol{m} = \alpha m_2 \cdots \mapsto \underline{\mathrm{T}}_{[\alpha]}(\boldsymbol{m}) = m_2 m_3 \cdots .$$

Ces définitions s'étendent de façon naturelle aux ensembles ou suites de mots.
Exemple. Considérons l'ensemble X à deux éléments $X = \{abac\dots,\ babc\dots\}$ sur l'alphabet $\mathcal{M} = \{a,\ b,\ c\}$. On calcule les décalés

$$\underline{\mathrm{T}}(X) = \{bac\dots,\ abc\dots\}, \quad \underline{\mathrm{T}}_{[a]}(X) = \{bac\dots\}, \quad \underline{\mathrm{T}}_{[b]}(X) = \{abc\dots\}, \quad \underline{\mathrm{T}}_{[c]}(X) = \varnothing.$$

[1] Le terme digital fait référence au fait que l'on décompose la donnée en «digits».

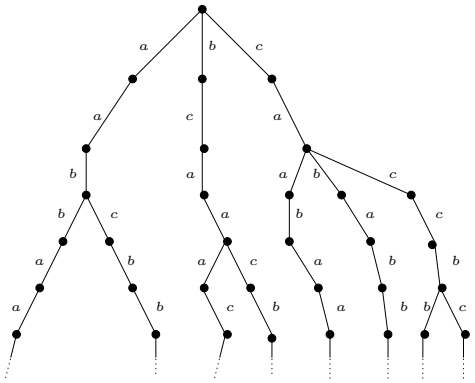


FIG. 2.1 – Arbre digital partiellement développé pour les mots $aabbaa...$, $aabcbb...$, $bcaaac...$, $bcaacb...$, $caabaa...$, $cababb...$, $caccbb...$, $caccbc...$.

2.1.2 Arbre digital

Nous définissons maintenant les structures d'arbre digital et de trie à l'aide de ce formalisme directement issu des systèmes dynamiques (le lien sera fait au chapitre 4).

DÉFINITION 2.1 (ARBRE DIGITAL). *Soit X une multipartie (avec répétitions possibles) finie d'éléments de $\mathcal{M}^{\mathbb{N}}$. On construit l'arbre digital associé à X, noté* $\mathsf{digital}(X)$*, grâce à aux règles récursives :*
– *si* $|X| = 0$*, alors* $\mathsf{digital}(X) = \varnothing$ *(l'arbre est vide).*
– *si* $|X| > 0$*, l'arbre* $\mathsf{digital}(X)$ *est*

$$\mathsf{digital}(X) = \left\langle \bullet, \mathsf{digital}(\underline{\mathrm{T}}_{[1]}(X)), \ldots, \mathsf{digital}(\underline{\mathrm{T}}_{[r]}(X)) \right\rangle,$$

où le symbole $\bullet$ désigne un nœud interne.

La structure d'arbre digital correspond à la vision la plus intuitive d'un ensemble de mots sous forme d'un arbre : on associe à chaque mot une branche de l'arbre. L'exemple de la figure 2.1 illustre en particulier le fait qu'un arbre digital construit sur des mots infinis est lui-même infini.

2.1.3 Trie

La structure de trie [48, 63, 71, 90] utilise une règle de terminaison récursive différente de celle de l'arbre digital. Elle sépare également les mots selon leurs préfixes, mais pour le trie on stoppe la récursion dès que tous les mots ont été distingués.

DÉFINITION 2.2 (TRIE). *On construit le trie associé à X, noté* $\mathsf{trie}(X)$ *grâce aux règles récursives suivantes :*
– *si* $|X| = 0$*, alors* $\mathsf{trie}(X) = \varnothing$ *est* vide.
– *si* $|X| = 1$ *(i.e. $X = \{\boldsymbol{x}\}$), alors* $\mathsf{trie}(X)$ *est un* nœud externe *étiqueté par $\boldsymbol{x}$.*
– *si* $|X| \geq 2$*, le trie* $\mathsf{trie}(X)$ *est*

$$\mathsf{trie}(X) = \left\langle \bullet, \mathsf{trie}(\underline{\mathrm{T}}_{[1]}(X)), \ldots, \mathsf{trie}(\underline{\mathrm{T}}_{[r]}(X)) \right\rangle,$$

où le symbole $\bullet$ désigne un nœud interne.

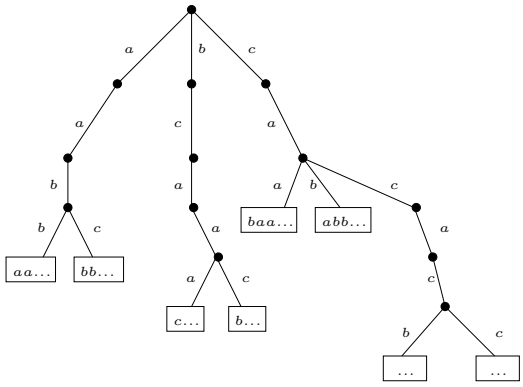


FIG. 2.2 – Trie construit sur le même ensemble de mots que celui de la figure 2.1.

Le trie correspondant au même ensemble de mots que pour la figure 2.1 est dessiné sur la figure 2.2. Le passage de l'arbre digital au trie permet de ne pas «développer» complètement la structure d'arbre.
La fonction essentielle reste de séparer les éléments de X d'après leurs préfixes. Le nombre de nœuds externes du trie est égal au nombre d'éléments de l'ensemble X (pourvu que les éléments de X soient tous distincts).
Au vu de ces définitions, on peut faire plusieurs remarques. Le trie comme l'arbre digital ne dépend que de l'ensemble X et non pas de l'ordre dans lequel on insère les éléments. Ensuite, si les éléments de X ne sont pas tous distincts, le trie est infini puisqu'on ne peut séparer deux mots identiques en examinant leurs préfixes. La branche correspondante se développe indéfiniment. Il est également intéressant de noter que le degré du trie (c'est à dire le nombre maximal de branchements d'un nœud du trie) est au plus égal à $\min(\mathrm{card}\ \mathcal{M}, n)$, où $\mathcal{M}$ est l'alphabet et n le cardinal de l'ensemble X. Si $\mathcal{M} = \{0, 1\}$ (cas binaire), le trie résultant sera binaire.

Dans la pratique, les mots sont finis. Mais le trie peut tout de même être construit grâce aux règles récursives de la définition dès lors qu'aucun mot n'est le préfixe d'un autre (par exemple, l'adjectif «uni» est un préfixe du nom commun «unicité»). Afin d'obtenir une structure universelle, on choisit généralement d'ajouter un caractère de terminaison (distinct de tout symbole de l'alphabet $\mathcal{M}$) à la fin de chacun des mots (en considérant l'ensemble $\{\texttt{uni\#}, \texttt{unicité\#}\}$ au lieu $\{\texttt{uni}, \texttt{unicité}\}$ par exemple). La propriété qu'aucun mot ne soit préfixe d'un autre est alors automatiquement vérifiée.

Remarque. La structure d'arbre binaire de recherche (qu'on étudiera de façon approfondie au chapitre 6) est très différente de la structure de trie. L'ordre d'insertion des clés devient primordial. Le principe de construction d'un arbre binaire de recherche est le suivant : on insère un à un les éléments d'une liste de clés distinctes. On a pour une liste $X = (x_1, x_2, \ldots, x_n)$ de clés
– si $|X| = 0$, alors $\mathsf{abr}(X) = \varnothing$ est *vide*.
– si $|X| > 0$, alors $\mathsf{abr}(X)$ est

$$\mathsf{abr}(X) = \langle x_1, \mathsf{abr}(X_{<x_1}), \mathsf{abr}(X_{>x_1}) \rangle\,,$$

où $X_{<\alpha}$ désigne la liste des clés de X strictement inférieures à α.

2.1.4 Paramètres

La structure de trie permet de différencier un ensemble de mots selon leurs préfixes. Ainsi un nœud interne n du trie est en correspondance directe avec le préfixe commun à tous les mots

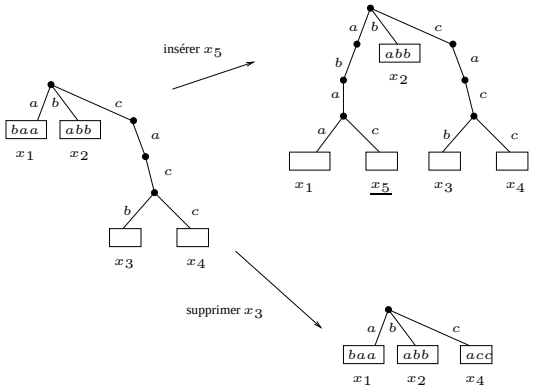


FIG. 2.3 – À gauche, le trie construit sur l'ensemble $X = \{x_1, x_2, x_3, x_4\}$, avec $x_1 = abaa$, $x_2 = babb$, $x_3 = cacb$, $x_4 = cacc$. À droite, les trie construits sur $X \setminus \{x_3\}$ et $X \cup \{x_5\}$ avec $x_5 = abac$.

stockés (il y en a au moins deux par définition) dans le sous-arbre de racine n. Ce préfixe n'est autre que la concaténation des lettres étiquetant la branche menant de la racine jusqu'au nœud n.

La profondeur d'un nœud est le nombre de liens le séparant de la racine. La profondeur d'un nœud est donc aussi la longueur du préfixe correspondant à ce nœud.

Il existe plusieurs paramètres naturels pour les tries qui permettent d'analyser le comportement d'algorithmes utilisant ce type de structure. Trois paramètres sont essentiels dans les tries :

- La *hauteur* H du trie est la profondeur maximale atteinte parmi l'ensemble des nœuds du trie. C'est aussi la longueur maximale d'une branche du trie, et donc le nombre maximal de comparaisons nécessaires pour départager deux mots. La hauteur est aussi le maximum des longueurs d'un préfixe partagé par au moins deux mots du trie.
- La *taille* T du trie est le nombre de nœuds internes du trie. Ce paramètre est lié à la taille mémoire nécessaire au stockage du trie, le nombre total de nœuds étant égal à la somme de la taille et du nombre n de nœuds externes. Rappelons qu'un nœud interne du trie existe dès lors que le mot associé est préfixe d'au moins deux mots du trie.
- La *longueur de cheminement externe* L est la somme des profondeurs des nœuds externes. Ce paramètre mesure le nombre de comparaisons nécessaires à la construction du trie, ou pour différencier tous les mots. En divisant par n, le nombre de nœuds externes, on obtient la profondeur moyenne d'un nœud externe. Un nœud externe du trie correspond au préfixe $\boldsymbol{w}$ d'un unique élément de l'ensemble et ce préfixe est de plus minimal : si on en enlève le dernier caractère, le mot obtenu devient préfixe d'au moins deux éléments du trie.

Exemple. Pour le trie de la figure 2.2, les valeurs des paramètres sont de 6 pour la hauteur, 13 pour la taille et 36 pour la longueur de cheminement.

2.1.5 Opérations usuelles

Insertion et suppression. Nous avons déjà remarqué que la structure de trie ne dépend pas de l'ordre dans lequel les mots sont insérés. Pour chaque insertion, on va créer exactement un nœud externe mais aussi un certain nombre de nœuds internes, ce nombre pouvant varier énormément. Symétriquement la suppression d'un mot correspond à la suppression d'un nœud externe et de nœuds internes qui après la modification du trie ne sont plus valides (dans le cadre de la définition 2.2). Ce type de phénomène est représenté sur la figure 2.3.

Recherche. Le principe de la recherche est le suivant : pour tester si un mot $\boldsymbol{w}$ est présent parmi un ensemble de mots X, on commence par construire le trie associé à l'ensemble X. Ensuite, on parcourt le mot $\boldsymbol{w}$ lettre par lettre en empruntant la branche correspondante du trie jusqu'à un blocage éventuel (c'est à dire jusqu'à ce que la lettre sur le flot d'entrée ne corresponde à aucun fils du nœud courant). S'il y a blocage sur un nœud interne, le mot n'est pas présent. Si on arrive à un nœud externe, il reste à comparer la fin du mot et les symboles stockés sur ce nœud externe. Cette utilisation typique d'un trie peut également être vue sous l'angle des automates. La structure d'arbre est alors un automate associé à l'ensemble X qui décide si un mot est présent ou non dans cet ensemble.

2.2 Généralisations

La structure de trie admet plusieurs variations qui s'appuient toujours sur une représentation des données sur un alphabet.

2.2.1 Bucket trie

La première généralisation concerne le bucket trie, encore appelé b-trie. C'est une structure de trie dont on a relâché la règle récursive d'arrêt lors de la construction.

DÉFINITION 2.3 (BUCKET TRIE). *Soit $b \geq 1$ un entier appelé* capacité. *Soit X une multipartie (avec répétitions possibles) finie d'éléments de $\mathcal{M}^{\mathbb{N}}$. On définit le b-trie associé à X, noté* $\mathsf{bucket}(X)$ *grâce aux règles récursives suivantes :*
– *si* $|X| = 0$, *alors* $\mathsf{bucket}(X)$ *est* vide.
– *si* $|X| \leq b$, *alors* $\mathsf{bucket}(X)$ *est un* nœud externe *étiqueté par* X.
– *si* $|X| > b$, $\mathsf{bucket}(X)$ *est*

$$\mathsf{bucket}(X) = \left\langle \bullet, \mathsf{bucket}(\underline{\mathrm{T}}_{[1]}(X)), \ldots, \mathsf{bucket}(\underline{\mathrm{T}}_{[r]}(X)) \right\rangle,$$

où le symbole $\bullet$ désigne un nœud interne.

On voit immédiatement que le cas du trie usuel correspond au cas $b = 1$. La quantité b est appelée capacité, car on peut voir les nœuds externes comme des pages ayant la «capacité» de stocker jusqu'à b données. Un exemple de b-trie avec $b = 3$ est dessiné sur la figure 2.4.
Le principe de recherche est analogue à celui d'un trie classique. Pour rechercher $\boldsymbol{w}$, on chemine dans le b-trie en suivant la branche dictée par les lettres de $\boldsymbol{w}$, jusqu'à atteindre une page. Il reste alors à vérifier que la clé est bien présente dans cette page. Le parallèle avec un dictionnaire est immédiat et naturel. On commence d'abord par rechercher la page par rapport aux premières lettres. Une fois la page localisée, une deuxième procédure de recherche, séquentielle ou dichotomique, est déclenchée.

Le b-trie est utilisable comme index (stocké en mémoire interne) pour de grands volumes de données. En ce cas, les sommets externes sont maintenus sur une mémoire secondaire (disque magnétique) dont la taille de page dicte la valeur de b.

2.2.2 Arbre PATRICIA

L'arbre PATRICIA (acronyme de *Practical Algorithm To Retrieve Information Coded In Alphanumeric*) a été introduit par Donald R. Morrison en 1968 [78]. L'idée est de contracter les branches en ne gardant que les nœuds internes avec au moins deux fils. En effet les nœuds internes sans

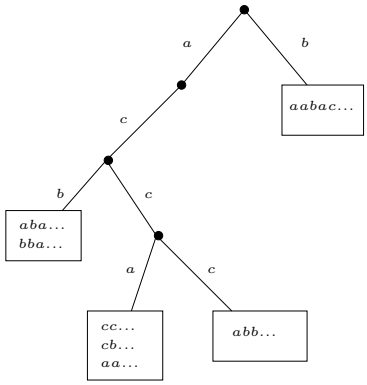


FIG. 2.4 – Un b-trie de capacité $b = 3$ pour les mots $acbaba...$, $acbbba...$, $accacc...$, $accacb...$, $accaaa...$, $acccbc...$, $baabc....$

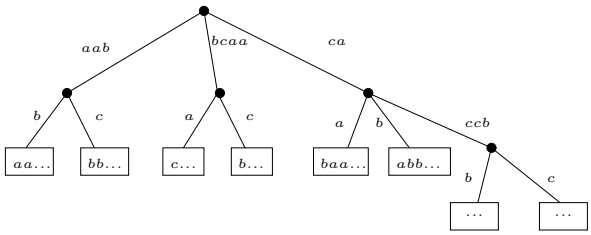


FIG. 2.5 – Un trie PATRICIA pour l'ensemble de mot de la figure 2.1.

branchement correspondent à des préfixes qui ne permettent pas de départager deux mots. Les liens de l'arbre ne sont donc plus étiquetés par des lettres mais par des mots. Dans une implantation, on peut stocker en chaque nœud un couple `(lettre,skip)` qui indique la première lettre de l'étiquette (c.-à-d. la lettre qui a mené à la création d'une nouvelle branche) et le nombre de symboles à «sauter» qui de toute façon n'influencent pas la recherche. De la sorte on accélère l'accès à la feuille susceptible de contenir l'information recherchée : on conclut par une comparaison lettre à lettre entre le motif recherché et le mot trouvé dans la mesure où certaines lettres ont pu être sautées. Une implantation de PATRICIA trie est présentée dans [90]. Un exemple de PATRICIA trie est dessiné sur la figure 2.5.

2.2.3 Arbre des suffixes

L'arbre des suffixes est une structure de données qui s'appuie sur une structure de trie. Ce trie est construit sur l'ensemble des suffixes d'un texte. On l'enrichit encore en ajoutant des liens transversaux, appelés liens «suffixes». Ces liens sont destinés à accélérer la recherche d'un sous-mot dans l'arbre. La structure obtenue est un *arbre des suffixes*.

Soit $\boldsymbol{t}$ un mot fini de $\mathcal{M}^*$ de longueur $|\boldsymbol{t}| = n$. L'ensemble des suffixes $\mathsf{suff}(\boldsymbol{t})$ est l'ensemble des décalés du mot $\boldsymbol{t}$

$$\mathsf{suff}(\boldsymbol{t}) = \{\underline{\mathrm{T}}^k(\boldsymbol{t}) \mid 0 \leq k \leq n-1\}.$$

Le *trie des suffixes* associé à un texte $\boldsymbol{t}$ est l'arbre $\mathsf{trie}(\mathsf{suff}(\boldsymbol{t}))$ construit sur l'ensemble des suffixes de $\boldsymbol{t}$. Cette structure fait correspondre les nœuds de l'arbre et les facteurs du texte. On utilise un tel arbre dans de nombreuses applications [3] comme la recherche de motifs (un tel arbre permet

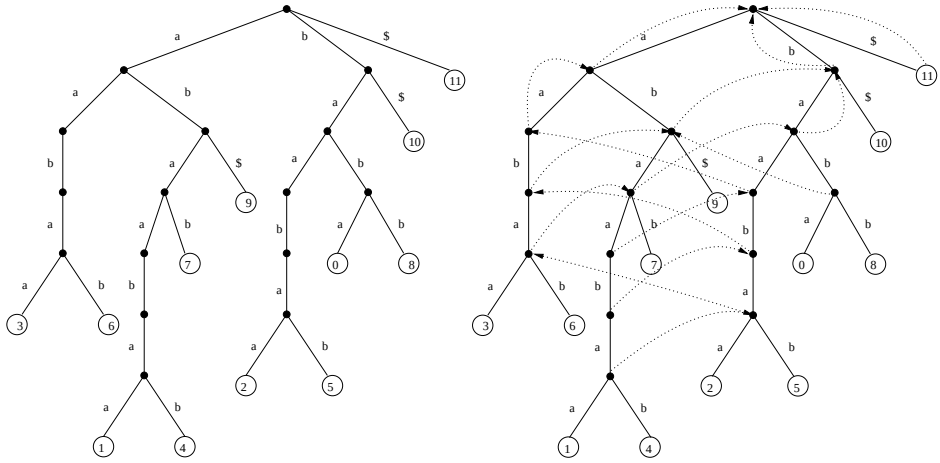


FIG. 2.6 – Trie des suffixes et arbre des suffixes correspondant pour le mot $babaabaabab\$$. Les liens suffixes sont en pointillé, et les feuilles contiennent l'indice du suffixe (une feuille etiquetée k correspond au k^{e} suffixe).

de chercher un mot $\boldsymbol{w}$ dans le texte $\boldsymbol{t}$ en un temps $O(|\boldsymbol{w}|)$, simplement en examinant le chemin dans le trie correspondant au mot $\boldsymbol{w}$) ou encore la compression. Cette structure sera abondamment utilisée dans la section 2.3 et constitue sans doute la structure d'arbre centrale dans les problèmes autour des mots.

Afin de simplifier la présentation de la structure, on suppose qu'aucun mot de l'ensemble des suffixes n'est le préfixe d'un autre. Il suffit par exemple que le texte se termine par un symbole de terminaison distinct de toutes les lettres apparaissant dans $\boldsymbol{t}$ pour que cette condition soit vérifiée.

La procédure de construction naïve d'un trie des suffixes, consistant à insérer les suffixes un à un, donne un algorithme en $O(n^2)$ (avec $n = |\boldsymbol{t}|$) dans le pire des cas. Du point de vue des applications, il y a alors deux options. Soit le trie des suffixes est utilisé par l'application comme une structure statique ou quasi-statique (cas du dictionnaire par exemple). La méthode naïve est alors parfaitement envisageable. Dans le cas d'une structure dynamique, vouée à être constamment modifiée, le coût quadratique dans le pire des cas peut être rédhibitoire (quoique ce coût se réduise à $O(n \log n)$ en moyenne [4]) et des méthodes de construction plus efficaces sont souhaitables.

De fait, plusieurs auteurs [75, 103, 98] proposent des méthodes de construction qui permettent de construire ce trie en temps linéaire, si l'on ajoute un *lien suffixe* en chaque nœud interne. L'arbre «enrichi» obtenu est appelé *arbre des suffixes*.

Pour un nœud interne du trie d'étiquette $a\boldsymbol{w}$ (où a est une lettre de l'alphabet et $\boldsymbol{w}$ un mot éventuellement vide), le lien suffixe pointe vers le nœud interne d'étiquette $\boldsymbol{w}$ (en prenant comme convention que le nœud interne associé au mot vide ϵ est la racine). La figure 2.6 montre le trie des suffixes et l'arbre des suffixes correspondant.

Bien sûr, de la même façon qu'il est possible de considérer la variante PATRICIA d'un trie, l'arbre des suffixes peut être compacté (voir exemple sur la figure 2.7). C'est naturellement cette structure de trie compacté qui est la plus utilisée en pratique. Nous en détaillons ici le mécanisme de construction pour l'algorithme de MacCreight, notamment la façon dont sont maintenus les liens suffixes.

Le but est d'obtenir un arbre qui, en chaque nœud interne, d'étiquette $a\boldsymbol{x}$, affecte un pointeur vers le nœud représentant $\boldsymbol{x}$, qui est le mot obtenu en supprimant le premier symbole a. L'idée

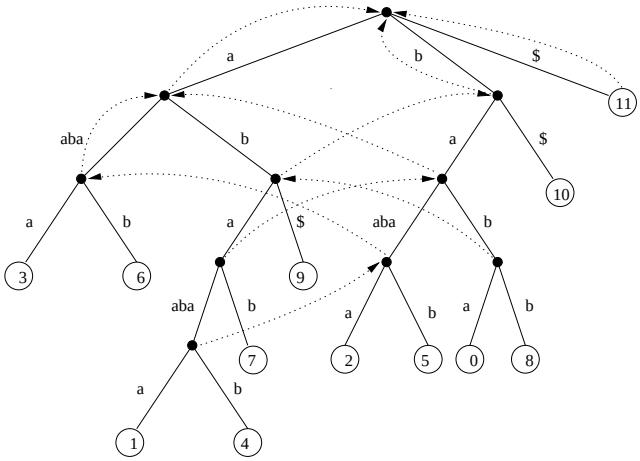


FIG. 2.7 – Arbre compact des suffixes pour le mot $babaabaabab\$$ (voir la figure 2.6).

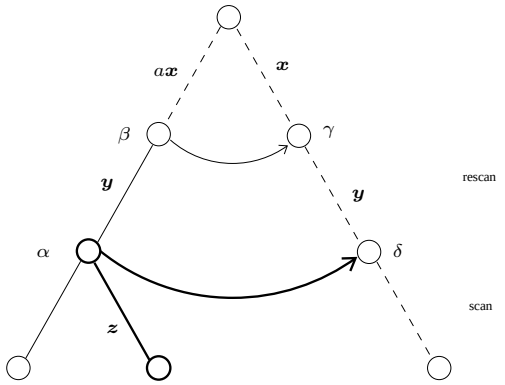


FIG. 2.8 – La construction d'un arbre des suffixes.

générale est que si le i^{e} suffixe $\underline{\mathrm{T}}^{i-1}(\boldsymbol{t})$ vient juste d'être inséré en un nœud de niveau d, il n'est pas nécessaire de retourner à la racine pour insérer le $(i+1)^{\text{e}}$ suffixe $\underline{\mathrm{T}}^{i}(\boldsymbol{t})$. Emprunter un lien suffixe proche raccourcit la recherche et ramène presque immédiatement sur la bonne branche de l'arbre.

La figure 2.8 montre de quelle façon les liens suffixes sont crées et utilisés. Supposons que nous cherchions à insérer un mot $\boldsymbol{w}$ dans le trie. Dans la figure 2.8, on suppose que $\boldsymbol{w}$ peut s'écrire $a\boldsymbol{xyz}$ où a est un caractère et $\boldsymbol{x}$, $\boldsymbol{y}$ et $\boldsymbol{z}$ sont des mots. En suivant la branche correspondante du trie, on reconnaît le mot $a\boldsymbol{xy}$ mais il y a échec pour le mot $\boldsymbol{z}$ (on parle de «mismatch» au niveau du premier caractère de $\boldsymbol{z}$). L'arbre étant contracté, il n'y a pas nécessairement de nœud interne qui corresponde à $a\boldsymbol{xy}$. Sur la figure 2.8, on considère le cas le plus compliqué où l'on crée un nouveau nœud interne α correspondant au mot $a\boldsymbol{xy}$. Le lien suffixe du nœud α doit alors être calculé. Nous insérons alors le suffixe suivant $\boldsymbol{xyz}$. On commence par remonter au nœud β (le père de α) qui représente le mot $a\boldsymbol{x}$, et on emprunte le lien suffixe jusqu'au nœud γ qui représente $\boldsymbol{x}$. On descend à nouveau dans l'arbre, d'abord en suivant la branche correspondant à $\boldsymbol{y}$ (phase dite de «rescanning» puisqu'on a déjà lu et recherché ce mot auparavant) jusqu'au nœud δ, puis à partir de δ la branche correspondant à $\boldsymbol{z}$ (phase dite de «scanning»). La première partie est appelée phase de rescanning car elle correspond à une portion du mot déjà vue au cours de l'insertion du

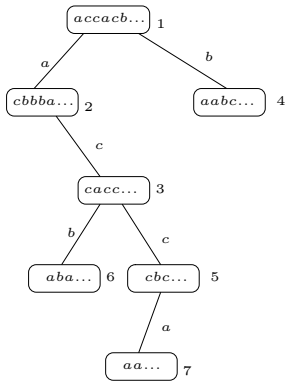


FIG. 2.9 – Un arbre digital de recherche obtenu en insérant successivement les mots $accacb...$, $acbbba...$, $accacc...$, $baabc...$, $acccbc...$, $acbaba...$, $accaaa...$.

précédent suffixe, et ne nécessite donc pas de vérifier que le chemin correspondant est valide. (De fait, éviter ces vérifications s'avère primordial pour obtenir un algorithme qui fonctionne en temps linéaire.) Cette phase peut s'achever sur un nœud interne δ existant. Dans le cas contraire, un nouveau nœud est créé. Ce nœud δ devient alors la destination du lien suffixe de α. La preuve de correction de l'algorithme repose sur le fait qu'a été rétabli l'invariant selon lequel tout nœud interne sauf éventuellement le dernier créé possède un lien suffixe valide.

2.2.4 Arbre digital de recherche

L'arbre digital de recherche allie le principe de construction d'un arbre binaire de recherche (un nœud interne contient une clé) et celui d'un trie (les lettres composant la clé guident la recherche). Cette structure se distingue du trie car elle stocke un mot en chaque nœud. Dans la littérature, l'arbre digital de recherche est la plus souvent un arbre binaire [63, 90]. On peut évidemment généraliser au cas d'un alphabet non binaire (ce qui est fait dans la suite).

DÉFINITION 2.4 (ARBRE DIGITAL DE RECHERCHE). *Les arbres digitaux de recherche sont construits sur une suite finie $S = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n)$ de n éléments de $\mathcal{M}^{\mathbb{N}}$. L'arbre digital de recherche* $\mathrm{dst}(S)$ *est défini récursivement par :*
– *Si $|S| = 0$ alors l'arbre digital de recherche* $\mathsf{dst}(S) = \varnothing$.
– *Si $|S| > 0$, l'arbre digital* $\mathsf{dst}(S)$ *de recherche est*

$$\mathsf{dst}(S) = \left\langle \boldsymbol{s}_1, \mathsf{dst}(\underline{\mathrm{T}}_{[1]}(S \backslash \boldsymbol{s}_1)), \ldots, \mathsf{dst}(\underline{\mathrm{T}}_{[r]}(S \backslash \boldsymbol{s}_1)) \right\rangle,$$

où pour $S = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n)$ la notation $S \backslash \boldsymbol{s}_1$ désigne $(\boldsymbol{s}_2, \ldots, \boldsymbol{s}_n)$.

L'arbre digital de recherche dépend de l'ordre d'insertion des éléments. Contrairement au trie, l'arbre est construit sur une suite de mots et non sur un ensemble. On remarque qu'un arbre digital construit sur n mots possède exactement n nœuds.

Le principe de recherche est le suivant : Soit $\boldsymbol{x}$ la clé de taille k à rechercher dans l'arbre $\mathrm{dst}(S)$. La clé $\boldsymbol{s}_1$ à la racine de $\mathrm{dst}(S)$ est comparée à $\boldsymbol{x}$. En cas de succès, la recherche est finie. Sinon, on va rechercher $\underline{\mathrm{T}}(\boldsymbol{x})$ (le mot $\boldsymbol{x}$ privé de son premier symbole) dans le sous-arbre relatif à la première lettre de $\boldsymbol{x}$. Il y a échec si le sous-arbre n'existe pas. La recherche d'une clé conduit à parcourir une branche qui est nécessairement un préfixe de la clé (comme pour un trie). La

différence avec le trie se situe dans le fait qu'on effectue une comparaison de mot (et non d'un symbole) en chaque nœud interne, et que l'on s'arrête en cas d'égalité.

2.2.5 Trie à k dimensions

Dans un trie à k dimensions, ou k-d trie, les éléments à considérer (l'univers des clés) ne sont plus des mots infinis mais des k-uplets de mots infinis. L'idée est d'associer à un k-uplet de mots $\overrightarrow{s} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_j)$ (où chaque mot $\boldsymbol{s}_i$ s'écrit $s_{i,1}s_{i,2}\cdots s_{i,k}\cdots$) un mot $\tilde{\boldsymbol{s}}$ construit en alternant les symboles des différentes composantes

$$\tilde{\boldsymbol{s}} = \underbrace{s_{1,1}s_{2,1}\cdots s_{k,1}}_{1^{\text{ers}}\text{ symboles}}\ \underbrace{s_{1,2}s_{2,2}\cdots s_{k,2}}_{2^{\text{es}}\text{ symboles}}\ \cdots\ \underbrace{s_{1,j}s_{2,j}\cdots s_{k,j}}_{j^{\text{es}}\text{ symboles}}\ \cdots .$$

Par exemple, le mot construit à partir du vecteur à trois dimensions

$$\overrightarrow{s} = \begin{pmatrix} \texttt{abaa}\cdots \\ \texttt{baab}\cdots \\ \texttt{abab}\cdots \end{pmatrix}$$

est le mot $\tilde{\boldsymbol{s}} = \texttt{aba bab aaa abb}\cdots$.
Le k-d trie est alors construit pour un ensemble $X = \{\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}\}$ de mots k dimensionnels en considérant le trie construit sur l'ensemble de mots (simples) $\tilde{X} = \{\tilde{x}_1, \ldots, \tilde{x}_n\}$ construits par ce procédé.

Cette structure est adaptée à la recherche partiellement spécifiée (*partial match query*). Une requête partielle consiste à rechercher un vecteur de mots (une donnée multidimensionnelle) dont certaines composantes sont non spécifiées. Par exemple, la requête $(\texttt{abc}, *, \texttt{def})$ correspond à une requête '`a.d b.e c.f`' dans le trie correspondant (où le '.' correspond à une lettre quelconque). Une recherche d'un motif avec des lettres indéterminées dans un trie consiste à considérer non plus une seule branche mais à une portion de l'arbre.

2.3 Applications

La structure de trie joue un rôle essentiel en algorithmique et ce, à plusieurs niveaux. Tout d'abord, le principe de partitionnement propre à la construction du trie, départager des mots en comparant leurs préfixes, est un principe facile à mettre en œuvre lorsque l'on traite des données digitales. Plusieurs algorithmes probabilistes l'utilisent et, bien souvent, la preuve de leur bon fonctionnement repose sur des propriétés d'un arbre digital qui est sous-jacent au déroulement de l'algorithme.

Ensuite, la structure de trie en tant que structure de données est réellement centrale dans bon nombre de problèmes. Il peut s'agir de la structure de trie décrite auparavant ou encore de légères variations comme l'arbre des suffixes ou encore les b-tries.

Le trie apparaît donc de deux façons différentes dans les applications : soit explicitement, en tant que structure de données, soit implicitement, lorsque seul le principe de partitionnement est utilisé.

2.3.1 Algorithmes sur les mots

Recherche de mots et de motifs. La recherche de motifs (ou encore «pattern matching») est un outil de base de l'informatique. Le problème de base consiste à déterminer si un motif a une occurrence dans un texte. La structure de trie est pleinement adaptée à ce type de problème.

La situation la plus simple où l'on cherche si un mot est présent dans un ensemble de mots est traitée immédiatement grâce à la recherche dans le trie.

Dans le cas de la recherche d'un motif $\boldsymbol{w}$ dans un texte $\boldsymbol{t}$, l'outil le mieux adapté est l'arbre des suffixes (voir section 2.2.3).

On peut aussi vouloir chercher plusieurs motifs (fixés) dans un texte (variable). On commence alors par construire un trie sur l'ensemble des motifs. Pour chaque position i dans le texte, on parcourt le trie à partir de la racine. Si on aboutit à un nœud externe, un des motifs est présent sinon on recommence à la position $i+1$ dans le texte. Cette méthode peut être affinée en introduisant une fonction de retour plus adaptée (afin de ne pas revenir en cas d'échec à la position $i+1$).

La structure de trie se prête bien à la recherche partielle (c'est cette caractéristique qui est utilisée pour les k-d tries). En effet, une requête contenant des lettres indéterminées ou contraintes dans un certain intervalle, peut se voir comme un parcours d'une partie du trie (et non plus d'une branche simple comme dans le cas d'une requête de base). Nous reviendrons sur cet aspect qui est un des points forts de la structure du ternary search trie au chapitre 5.

Problèmes sur les mots. On a examiné les utilisations les plus immédiates des tries sur un ensemble de mots. Nous présentons un catalogue (non exhaustif) lié à des problèmes de mots (voir [3] pour plus d'exemples). La structure d'arbre des suffixes est centrale dans bien des cas (voir l'article [3] d'Apostolico pour plus de précisions et références).

– *Recherche de la première, dernière ou toute occurrence d'un motif dans un mot.* Au cours de la construction de l'arbre des suffixes, on peut ajouter des informations sur les nœuds pour effectuer une recherche d'un type particulier. Dans l'arbre des suffixes on peut étiqueter les nœuds externes avec l'indice du suffixe qui a mené à sa création. En ajoutant pour chaque nœud interne un lien vers le nœud externe dont l'étiquette est minimale dans son sous-arbre, on trouve facilement la première occurrence de $\boldsymbol{w}$ dans un texte en $O(|\boldsymbol{w}|)$. On peut évidemment faire la recherche du dernier motif en temps linéaire de façon symétrique.
On parcourt les nœuds externes dans le sous-arbre correspondant à un mot $\boldsymbol{w}$, on obtient toutes les occurrences de $\boldsymbol{w}$ dans le texte $\boldsymbol{t}$.
– *Fréquence d'un motif dans un mot.* On peut construire un arbre des suffixes *pondéré* indiquant pour chaque nœud de l'arbre le nombre d'occurrences $C(\boldsymbol{w})$ du mot $\boldsymbol{w}$ correspondant. Il suffit pour cela de parcourir l'arbre de bas en haut en affectant pour chaque nœud la somme des poids de ses fils. On retrouve alors le poids $C(\boldsymbol{w})$ en temps $O(|\boldsymbol{w}|)$ en examinant le nœud correspondant à $\boldsymbol{w}$.
– *Identifiant d'une sous-chaîne.* L'identifiant d'une sous-chaîne pour une position i correspond pour un texte $\boldsymbol{t} = \boldsymbol{t}\langle 1, n\rangle$ à la sous-chaîne $\boldsymbol{t}\langle i, k\rangle$ la plus longue telle que $\boldsymbol{t}\langle i, k-1\rangle$ soit suffixe d'un autre suffixe de $\boldsymbol{t}$. Cela permet de savoir combien de lettres dans un motif sont nécessaires pour identifier complètement la position dans un texte $\boldsymbol{t}$ (voir [1]).
– *La plus longue sous-chaîne répétée d'un mot.* La plus longue sous-chaîne répétée d'un mot correspond dans l'arbre des suffixes non compacté au mot associé au nœud interne de profondeur maximale.
– *Le plus grand facteur commun de deux mots.* La construction d'un arbre des suffixes sur la chaîne $\boldsymbol{x}\#\boldsymbol{y}\$$ où $\#$ n'est pas un symbole de l'alphabet permet de trouver le plus grand facteur commun de $\boldsymbol{x}$ et $\boldsymbol{y}$ en temps $O(|\boldsymbol{x}| + |\boldsymbol{y}|)$.
– *Détection des carrés d'un mot.* Un carré de $\boldsymbol{t}$ est un mot de la forme $\boldsymbol{ww}$, où $\boldsymbol{w}$ est un mot *primitif*, c.-à-d. un mot qui ne peut s'exprimer comme une puissance $\boldsymbol{v}^k$ avec $k > 1$. En modifiant légèrement la procédure de construction de l'arbre des suffixes, il est possible de trouver tous les préfixes carrés d'un texte $\boldsymbol{t}$.
– *Statistiques sans recouvrement des facteurs d'un mot.* On peut calculer pour un texte $\boldsymbol{t}$ et un de ses facteurs $\boldsymbol{w}$ le nombre k d'occurrences distinctes de $\boldsymbol{w}$ tel qu'il soit possible d'écrire

$\boldsymbol{t} = \boldsymbol{w}_1\boldsymbol{w}\boldsymbol{w}_2\boldsymbol{w}\boldsymbol{w}_3 \cdots \boldsymbol{w}\boldsymbol{w}_{k+1}$ avec $\boldsymbol{w}_d$ pouvant être vide ($d = 1, 2, \ldots, k+1$). On peut tout à fait prendre en compte cela en stockant en chaque nœud ce nombre d'occurrences au fur et à mesure de la construction (cela est évidemment moins simple que d'établir les statistiques avec recouvrement).

Bioinformatique. Toutes ces utilisations se transposent évidemment dans le domaine de la bioinformatique. Une structure digitale comme l'arbre compacté des suffixes peut être utilisée pour l'analyse de séquences, l'identification de motifs «biologiquement» significatifs, la fabrication de séquence ou encore l'homologie entre séquences.
Il a également été proposé de comparer la quantité d'information «utile» entre plusieurs séquences en comparant la taille de ces séquences une fois compressées.

2.3.2 Compression, code de Huffman

Le principe de la compression consiste à utiliser une structure régulière dans un texte $\boldsymbol{t}$ pour en donner une représentation plus compacte. Nous ne nous intéressons dans les deux prochaines parties qu'à des techniques de compression conservative qui gardent toute «l'information» contenue dans le texte (par opposition à certaines techniques de compression d'image). Soient $\mathcal{M}$, Σ deux alphabets et $\boldsymbol{t}$ un texte construit sur $\mathcal{M}$. Compresser un message consiste à *coder* le message sur l'alphabet Σ (la plupart des applications utilisent un alphabet binaire $\Sigma = \{0, 1\}$ de sortie). Les notions de codage et de compression sont étroitement liées. Ainsi le but de la compression est d'obtenir un codage du texte qui soit le plus court possible.

Nous examinons deux techniques de compression qui ont un lien direct avec les tries : le codage «à la Huffman» (qui comprend le code de Huffman proprement dit, mais aussi le code de Fano-Shannon) et les techniques de compression «à la Lempel et Ziv» (dont nous présentons deux variantes LZ77 et LZ78) dans la section 2.3.3.

Si nous considérons un trie avec r nœuds externes (où r est le nombre de lettres différentes dans le texte), nous obtenons tout naturellement un code. Chaque symbole $i \in \mathcal{M}$ sera codé par le mot correspondant à la branche menant de la racine au nœud externe i. Par exemple, la figure 2.10 montre trois codes qui pourraient être utilisés pour coder `abracadabra`. Les codes obtenus sont respectivement

Codage 1 : `11 00 011 11 010 11 10 11 00 011 11`,
Codage 2 : `11011 0110 0111 11011 010 11011 11010 11011 0110 0111 11011`,
Codage 3 : `0 110 10 0 1111 0 1110 0 110 10 0`,

qui sont de longueurs respectives 25, 49 et 23.

La construction d'un code n'est autre que la détermination d'un «bon» trie de codage. On veut attribuer des codes courts aux symboles les plus fréquents et des codes plus longs aux symboles plus rares.

Une propriété importante du codage obtenu à partir d'un trie est qu'il est *sans préfixe*. Aucun mot binaire n'est le préfixe du code d'un autre symbole. (Cette propriété est évidente dès que l'on considère le trie sous-jacent du codage.)

Construction d'un arbre de codage

Huffman. Le *codage de Huffman* est une méthode (statique) pour construire le trie optimal pour la longueur du code produit. Elle construit l'arbre binaire par une démarche «bottom-up» de la manière suivante : au début on considère une forêt d'arbres binaires dont les éléments sont des nœuds externes correspondant aux symboles à coder. De plus, chaque nœud contient un champ `freq` qui a pour valeur la fréquence du symbole. La méthode est statique car les fréquences sont supposées connues à l'avance.

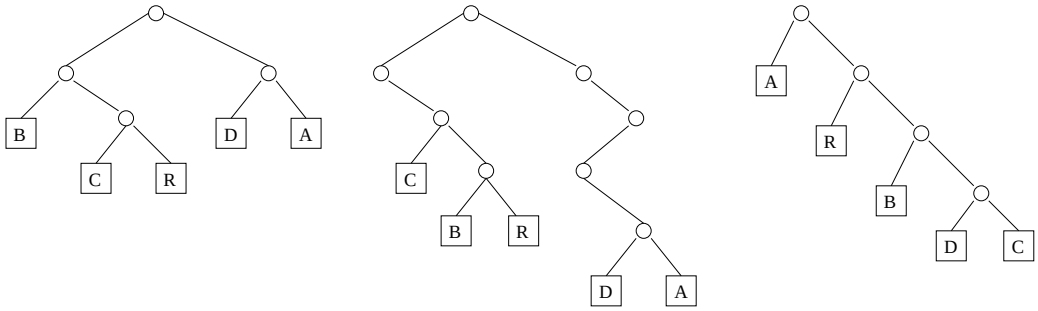


FIG. 2.10 – Trois tries permettant de coder les lettres A,B,R,C et D.

On choisit dans la forêt les deux nœuds qui possèdent les fréquences les plus faibles (à fréquence égale le choix d'un nœud ou d'un autre n'a pas d'importance). Un nouveau nœud interne dont les deux fils sont les deux nœuds choisis est construit. Le champ `freq` de ce nœud prend comme valeur la somme des fréquences de ses fils. On obtient une nouvelle forêt. Ce processus itératif s'arrête lorsque la forêt ne contient plus qu'un seul arbre qui est le trie de codage (voir la figure 2.11).

Le codage est optimal pour la modélisation proposée et parmi les codes sans préfixe. La longueur du code obtenu n'est autre que la longueur de cheminement externe pondérée du trie obtenu.

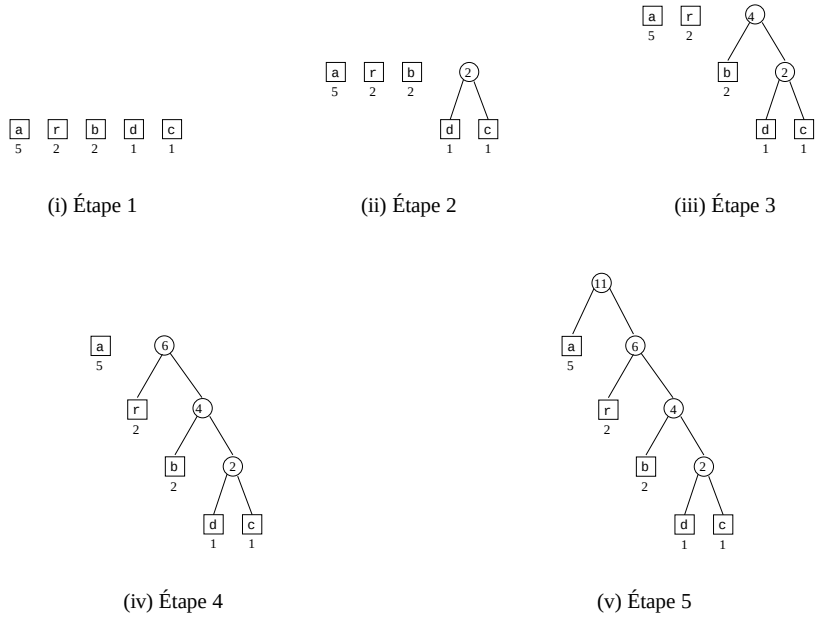


FIG. 2.11 – Les étapes de la construction du code de Huffman pour le texte `abracadabra`.

Fano-Shannon. Le codage de *Fano-Shannon* utilise une approche descendante qui est très proche du processus de construction d'un trie. La procédure est récursive. Les symboles de l'alphabet $\mathcal{M} = \{a_1, \ldots, a_r\}$ sont supposés rangés par ordre décroissant de probabilité ($p_1 \geq p_2 \geq \cdots \geq$

p_r). Soit $\boldsymbol{w} = a_1 a_2 \cdots a_r$ regroupant les lettres de l'alphabet. L'algorithme consiste à partitionner l'ensemble des symboles en deux sous-ensembles dont la somme des fréquences est sensiblement égale.

On construit l'arbre de codage successivement en construisant une suite d'arbres (A_i) :
– Au début, A_0 est composé d'un seul nœud portant l'étiquette $\boldsymbol{w} = a_1 a_2 \cdots a_r$.
– On applique récursivement sur le nœud externe courant les deux règles récursives de construction de l'arbre. Deux cas se présentent :
  (i) L'étiquette associée au nœud externe est composée d'une seule lettre. La récursion s'arrête. Le code de la lettre est le chemin menant de la racine à ce nœud externe.
  (ii) L'étiquette du nœud externe est un mot d'au moins deux lettres $\boldsymbol{w}\langle i, j\rangle$ ($i < j$) où $\boldsymbol{w}\langle i, j\rangle = \boldsymbol{w}\langle i\rangle\, \boldsymbol{w}\langle i+1\rangle \cdots \boldsymbol{w}\langle j\rangle$. On crée un nouveau nœud interne ayant pour fils gauche et fils droit deux nœuds externes ayant pour étiquettes respectives les mots $\boldsymbol{w}\langle i, k\rangle$ et $\boldsymbol{w}\langle k+1, j\rangle$, où l'entier k est déterminé par la formule

$$k = \min\{\ell \mid \sum_{n=i}^{\ell} p_n \geq \frac{1}{2} \sum_{n=i}^{j} p_n\}.$$

  Le processus de construction se poursuit récursivement sur chacun des deux nœuds externes crées.

La procédure suit quasiment les règles récursives de la construction d'un trie. Par construction, le code construit attribue les mots les plus longs aux symboles les moins probables.
Cette procédure produit un code sans préfixe qui n'est pas forcément optimal. Cependant en pratique, les deux méthodes produisent des longueurs de code équivalentes.

Codage et décodage

Une fois le trie obtenu, il suffit pour encoder un message de coder chaque symbole par le mot binaire relatif à la branche menant au nœud externe correspondant. Pour décoder, il suffit de suivre dans le trie à partir de la racine la branche relative au mot binaire à décoder (à gauche pour 0 et à droite pour 1) et d'émettre le symbole correspondant lorsque l'on atteint un nœud externe. On recommence ce processus avec le reste du mot binaire en repartant à nouveau de la racine.

2.3.3 Compression, Lempel-Ziv

Les techniques de compression à la Lempel et Ziv utilisent des méthodes à base de dictionnaires. C'est donc tout naturellement que la structure de trie est centrale pour ce type de compression (voir l'article de *survey* [6]).

Principe. Le compresseur comme le décompresseur maintiennent en parallèle un dictionnaire. Cette méthode est donc dynamique et son principe se résume grâce au schéma de la figure 2.12.

On parcourt le texte et au fur à mesure de la lecture du texte en ajoutant au dictionnaire des mots choisis. On ne transmettra alors pour les prochaines occurrences de ces mots que leurs références dans le dictionnaire.

Il existe de nombreuses variantes de l'algorithme de Lempel et Ziv. Le principe général est dans un premier temps de découper le texte en segments. On transmet pour chaque nouveau segment la référence dans le dictionnaire du mot qui correspond le mieux. C'est dans la stratégie de mise à jour du dictionnaire que se situe la différence majeure entre les deux méthodes données par Lempel et Ziv nommées LZ77 et LZ78. La première méthode (LZ77) insère dans le dictionnaire l'ensemble des suffixes du nouveau segment (ce qui revient à dire que le dictionnaire contient l'ensemble des facteurs du texte déjà vu) tandis que la deuxième (LZ78) n'insère que le segment lui-même dans le dictionnaire. Les algorithmes L77 et L78 sont donc définis par :

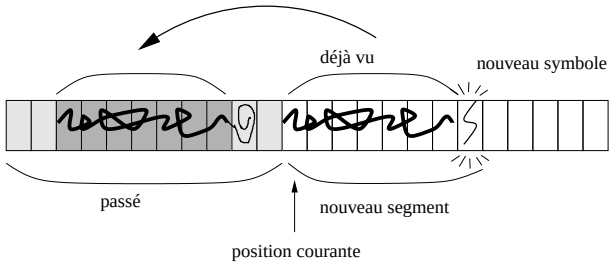


FIG. 2.12 – Principe de compression à la Lempel et Ziv : le texte est découpé en segment. Chaque nouveau segment est une portion de texte déjà vue auquel on ajoute le symbole suivant.

> LZ77. À partir de la position courante pour un nouveau segment s, on recherche le plus long *facteur* dans le texte déjà parcouru. On transmet la référence (position et longueur) du facteur déjà rencontré ainsi que la lettre suivante. Le nouveau segment crée est donc la concaténation du facteur trouvé et de la lettre. Le dictionnaire est mis à jour en ajoutant tous les suffixes du segment.
>
> LZ78. À partir de la position courante pour un nouveau segment s, on recherche le plus long *segment* déjà rencontré auparavant (et donc dans le dictionnaire). On transmet la référence (rang dans le dictionnaire) du segment déjà rencontré ainsi que la lettre suivante. Le nouveau segment s ainsi construit est donc la concaténation du segment trouvé et de la lettre. On ajoute le segment s au dictionnaire.

Par exemple, étant donné un texte ne contenant qu'une longue suite de a, les segments correspondants sont

LZ77 : `| a | aa | aaaa | aaaaaaaa |` ...
LZ78 : `| a | aa | aaa | aaaa | aaaaa |` ...

En d'autres termes, LZ77 définit comme nouveau segment le plus grand facteur possible dans ce qui a déjà été lu, même si cela recouvre plusieurs segments. L'algorithme LZ78 respecte les limites entre segments.

Pour LZ77, la transmission de chaque segment se fait grâce à un triplet du type `<p,l,c>` (position de l'occurrence reconnue, longueur du facteur, et bien sûr le nouveau symbole). Pour LZ78, on transmet un couple `<r,c>` (le rang du segment trouvé dans le dictionnaire et le symbole à ajouter pour créer un nouveau segment). La reconstruction du texte est simple puisqu'il suffit de réaliser «l'expansion des références».

Par exemple, les deux suites

LZ77 : `<0,0,a> <0,0,b> <0,0,r> <1,1,c> <1,1,d> <1,4,#>`,
LZ78 : `<0,a> <0,b> <0,r> <1,c> <1,d> <1,b> <3,a> <0,#>`,

encodent le texte «`abracadabra`#» (avec un symbole de terminaison #).

Chaque algorithme s'appuie sur une structure de donnée spécifique.

(i) **LZ77.** La structure d'arbre des suffixes est adaptée car elle permet de stocker l'ensemble des facteurs du texte. Si l'algorithme LZ77 originel (utilisant une fenêtre coulissante) est utilisé. Trois arbres des suffixes construits et maintenus de façon à suivre la fenêtre permettent de conserver un comportement linéaire (voir [85] pour l'explication de cette méthode).

(ii) **LZ78.** Le fonctionnement de l'algorithme de codage peut se voir sur un arbre digital de recherche (du moins dans le cas d'un alphabet binaire). En effet, considérons l'exécution de

l'algorithme LZ78 lorsque les k premiers caractères du texte $\boldsymbol{t}$ ont été lus. Le nouveau segment commence à la position $k+1$. On insère alors dans l'arbre digital de recherche le préfixe $\boldsymbol{p}$ de $\underline{\mathrm{T}}^k(\boldsymbol{t})$ minimal non présent dans l'arbre. Le dernier caractère de $\boldsymbol{p}$ est le caractère dont la lecture provoque la création d'un nouveau nœud (voir figure 2.13). La taille de l'arbre construit (c.-à-d. le nombre de nœuds) est alors la taille du texte compressé (ou encore le nombre de paires `<rang, symbole>` du codage).
En pratique, pour un alphabet quelconque, on utilise un arbre digital avec des nœuds possédant un champ `rang`. La procédure de codage et de construction de l'arbre se fait de la façon suivante. On part d'un arbre réduit à un nœud, la racine, dont le champ `rang` est initialisé à 0. Supposons que nous ayons déjà inséré i phrases et que nous soyons arrivés à la position $k+1$ du texte $\boldsymbol{t}$. Il reste donc à coder $\underline{\mathrm{T}}^k(\boldsymbol{t})$. On va construire la $(i+1)^{\mathrm{e}}$ phrase en parcourant le texte à partir de la position courante tout en suivant la branche correspondante dans l'arbre jusqu'à obtenir une situation d'échec en un nœud α sur une lettre a (situation de «mismatch», c.-à-d. qu'il n'y a aucune branche partant de α qui corresponde au symbole courant noté a). On crée alors un nouveau nœud β avec un champ `rang` égal à $i+1$. Pour le codage il suffit de sortir la paire `<r,` a`>` où `r` est la valeur du champ `rang` du nœud α et a est l'étiquette du lien qui vient d'être crée reliant les nœuds α et β (voir figure 2.13 et figure 2.14).

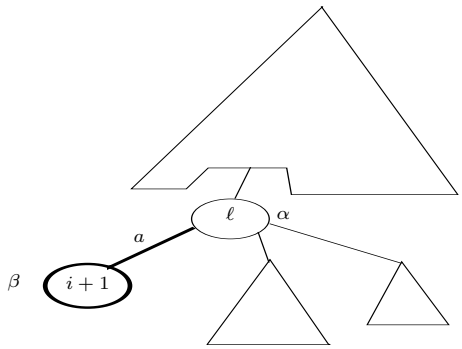


FIG. 2.13 – Création d'un nouveau nœud β au cours de l'insertion de la $(i+1)^{\mathrm{e}}$ phrase. On a reconnu la ℓ^{e} phrase suivie du symbole a. Le codage produit est le champ `rang` du nœud α, père de β, suivi du nouveau symbole, c.-à-d. `<`ℓ`,` a`>`.

D'autres techniques de compression ont un rapport soit avec la structure de trie soit avec les sources présentées au chapitre 4.

– *Codage arithmétique.* Le codage arithmétique utilise une approche complètement différente des deux précédentes pour effectuer la compression. Il consiste à coder un texte $\boldsymbol{t}$ par l'intervalle fondamental correspondant. Si cela n'a pas vraiment de rapport avec les tries proprement dits, cela en a avec les modèles de sources (en particulier les sources sans mémoire et les sources Markoviennes) utilisées dans le cadre de l'analyse des tries dans cette thèse (voir chapitre 4).
– *Gestion des codes d'échappement.* Toutes les méthodes de codage basées sur la modélisation statistique (Huffman, Fano-Shannon, compression arithmétique) peuvent considérer des modélisations de source d'ordre supérieur. La plupart du temps, afin de ne pas saturer la mémoire, cette modélisation prend en compte des «contextes» et des codes d'échappement afin de ne pas allouer de mémoire pour un contexte non vu. La structure de trie se prête naturellement pour rassembler les statistiques nécessaires à la modélisation.

2.3.4 Hachage dynamique

Les techniques de hachage permettent de rechercher un élément dans un grand volume de données de manière efficace. Une fonction de hachage associe à un élément u d'un ensemble U une clé de hachage à valeur dans X, où l'ensemble X est beaucoup plus petit que U. On construit une table de hachage qui sera utilisée de la façon suivante. Une fonction de hachage calcule pour chaque élément une clé de hachage. Cette clé de hachage est ensuite utilisée comme une adresse (ou un indice) dans la table de hachage, comme point de départ à partir duquel on cherche la clé proprement dite.

Dans un contexte où l'on utilise des unités de stockage externes (disques, bandes), la clé de hachage pointe vers une page qui ne peut stocker qu'un certain nombre d'éléments (on parle de capacité de la page). Au fur et à mesure que les pages se remplissent, de nouvelles pages doivent être allouées et les clés ne sont plus accessibles directement.

Plusieurs techniques tentent de remédier à cette situation : le hachage extensible (Fagin, Nievergelt, Pippenger, and Strong [25]), le hachage dynamique (Larson [66]) et le hachage virtuel (Litwin [69]). Le principe commun à toutes ces méthodes est que la fonction de hachage s'adapte localement et dynamiquement. Lorsque les données changent (à cause de suppressions et d'insertions), on conserve alors des accès quasi-directs. Ces trois techniques sont en général regroupées sous le terme de hachage dynamique.

Il existe des liens très importants entre la recherche digitale et les techniques de hachage dynamique. Ainsi, ces techniques font appel à une fonction de hachage H qui associe à chaque élément une clé de hachage binaire infinie. On accède à un ensemble de clés S au travers d'un b-trie construit sur les clés de hachage $H(S)$. Ainsi, les opérations effectuées sur une table de hachage dynamique avec une taille de page égale à b peuvent se voir comme des opérations sur un b-trie.

Le b-trie se décompose en deux parties : l'arbre formé par l'ensemble des nœuds internes constitue le *répertoire* tandis que les nœuds externes (ou *pages*) contiennent l'information proprement dite. Les trois variantes diffèrent surtout selon la façon de représenter ce répertoire : une structure chaînée pour le hachage dynamique, un arbre complet pour le hachage extensible ou encore une table binaire pour le hachage virtuel (qui diffère également par d'autres aspects).

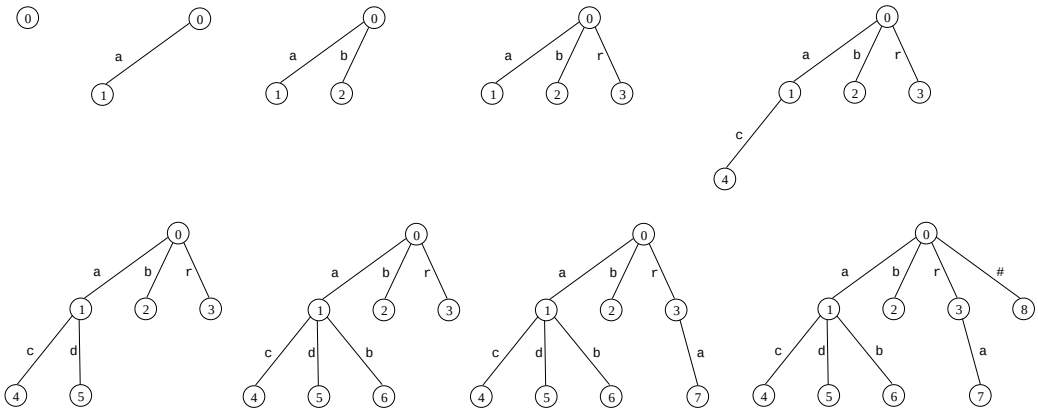


FIG. 2.14 – Les étapes correspondant au codage de `abracadabra#` pour l'algorithme LZ78 avec un arbre digital (avec en plus l'arbre initial réduit à un seul nœud). Les phrases insérées sont `a`, `b`, `r`, `ac`, `ad`, `ab`, `ra`, # et les couples correspondants sont `<0,a>`, `<0,b>`, `<0,r>`, `<1,c>`, `<1,d>`, `<1,b>`, `<3,a>`, `<0,#>`.

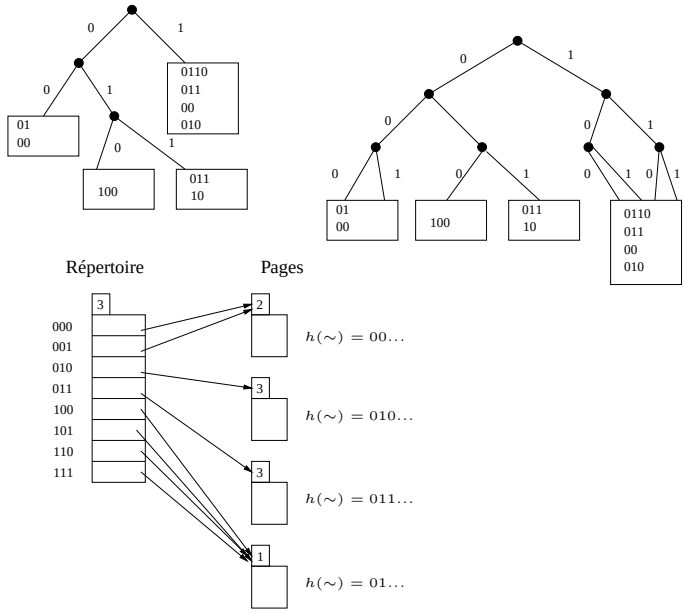


FIG. 2.15 – Le b-trie, l'arbre parfait en résultant et la table utilisée pour le hachage extensible pour l'ensemble de chaînes binaires $\{0001, 0000, 010100, 011011, 01110, 10110, 1011, 100, 1010\}$ pour une capacité $b = 4$.

On s'attache plus particulièrement au cas du hachage extensible, simple à implanter. On voit sur la figure 2.15 un b-trie, l'arbre complet correspondant et le tableau utilisée dans le cadre du hachage extensible. Le paramètre intéressant ici est la profondeur du répertoire. On appelle profondeur d du répertoire la hauteur «interne» du b-trie (la profondeur maximale d'un nœud interne du b-trie). La taille du tableau stockant le répertoire, également appelée taille du répertoire, est en effet $2^{d+1} - 1$. Il est alors facile d'accéder à une page donnée, il suffit d'interpréter les d premiers bits de la clé de hachage comme une adresse et de suivre le pointeur stocké à cette adresse dans le tableau (voir figure 2.15).

En cas de dépassement de la capacité d'une page, la taille du répertoire (ici la taille du tableau le représentant) est doublée et le nouveau tableau est calculé par un algorithme linéaire à partir du tableau initial (voir [25] pour les détails). Dans une implantation de cette méthode on associe à chaque page la profondeur dans le b-trie (appelée profondeur locale dans [25]) de cette page. Cela facilite les mises à jour. La bonne uniformité de la fonction de hachage garantit un bon taux de remplissage des pages.

Enfin, l'organisation des données dans chaque page est indépendante de la structure globale, mais il est souvent judicieux d'utiliser une table de hachage (en utilisant les s derniers bits de la clé de hachage, s étant une constante fixée).

L'analyse en moyenne de du hachage extensible s'attache au paramètre de profondeur du répertoire et de taille du répertoire ([33]). Le fait de ne pas considérer le b-trie mais l'arbre complet correspondant donne une taille du répertoire non linéaire.

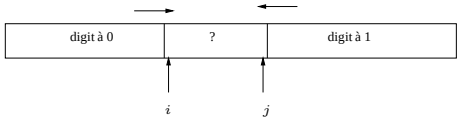


FIG. 2.16 – Principe du tri radix : L'ensemble des mots est partitionné selon le premier bit.

```
void radixsort(int l, int r, int b, char **s) {
  int i,j;
  char *t;
  if ((r>l) && (b>=0)) {
    i=l;
    j=r;
    do {
      while ((digit(s[i],b)==1) && (i<j)) i++;
      while ((digit(s[j],b)==0) && (i<j)) j--;
      t=s[i]; s[i]=s[j]; s[j]=t;
    } while (i<j);
    if (digit(s[r],b)==0) j++; /* si jamais on a que des 0 */
    radixsort(l,j-1,b+1);
    radixsort(j,r,b+1);
  }
}
```

FIG. 2.17 – Code C implantant le tri radix.

2.3.5 Utilisation du principe de partitionnement

Pour les algorithmes suivant, l'idée est toujours la même et se calque sur la construction d'un arbre digital (mais pas forcément un trie au sens strict du terme).

Tri radix

Cette technique de tri (parfois appelé *bucket sort*) est récursive et permet de trier lexicographiquement les données. Nous nous restreignons ici à un alphabet binaire. Le cas d'un alphabet de cardinal supérieur à deux est traité bien plus efficacement par un tri radix dont le principe se calque sur la structure de *ternary search trie*, ce qui fera l'objet de la section 5.3.4.

L'algorithme est dans son principe relativement proche de l'algorithme de tri rapide. Le code C correspondant est présenté dans la figure 2.17. De même qu'une exécution de tri rapide peut se «voir» sur un arbre binaire de recherche, le tri radix est lié à la structure de trie[2].

Supposons que nous puissions réorganiser les chaînes à trier afin que les chaînes commençant par 0 soient avant celles qui commencent par 1. Cela définit immédiatement une méthode récursive de tri : si les deux sous-ensembles de chaînes sont triés indépendamment, alors l'ensemble est trié. Ce principe est illustré par la figure 2.16. On parcourt les chaînes de gauche à droite afin de trouver une chaîne commençant par 1 ; on parcourt de droite à gauche pour trouver une chaîne qui commence par 0 ; on échange, et on continue jusqu'à ce que les deux pointeurs se croisent.

[2]La construction d'un trie à partir d'un ensemble X de chaînes de symboles permet grâce à une lecture des nœuds externes de gauche à droite de trier *lexicographiquement* cet ensemble.

Estimation de la cardinalité pour les bases de données

La cardinalité du multiensemble sous-jacent, c.-à-d. le nombre d'éléments *distincts*, revêt une importance fondamentale dans plusieurs algorithmes pour les bases de données. Cela nécessite d'avoir une méthode efficace pour calculer le cardinal de gros (multi)ensembles. La méthode naïve qui consiste à construire une liste des éléments sans répétitions est trop coûteuse au niveau des accès disque ou mémoire.

Comptage probabiliste. Pour résoudre ce problème, Flajolet et Martin [39] (voir également [77]) ont proposé l'algorithme suivant. Afin d'estimer la cardinalité N d'un ensemble $\mathcal{M}$ (avec répétitions), chaque élément $x \in \mathcal{M}$ est donné en argument à une fonction de hachage afin d'obtenir une chaîne binaire s de taille m. Dans l'algorithme, on va seulement considérer la chaîne binaire modifiée $\widetilde{s}$ construite en ne conservant que le premier bit à 1 en partant de la gauche par exemple.

Exemple. *Supposons que pour une donnée x la clé de hachage soit $s = 001001011$, la chaîne modifiée considérée est $\widetilde{s} = 001000000$.*

L'idée est alors de considérer R_N le rang maximal de 1 (toujours en partant de la gauche) parmi toutes ces chaînes. La valeur de R_N est alors une bonne approximation de $\log_2 N$. D'un point de vue informatique, cela consiste simplement à regarder la position du bit le plus significatif (en numérotant les bits à partir de 0 de gauche à droite) dans une chaîne binaire construite en composant bit à bit par un OU logique les chaînes modifiées.

L'analogie avec la construction d'un arbre digital binaire (qui n'est pas un trie puisque la condition récursive de terminaison n'est pas la même) est immédiate. Imaginons N personnes tirant à pile ou face, la liste des tirs d'une personne n'est autre que sa clé de hachage. Ceux qui tirent 1 s'arrêtent de jouer (cela correspond à suivre la branche droite de l'arbre digital) tandis que ceux qui tirent 0 rejouent (on suit alors la branche gauche). Le jeu s'arrête lorsque tout le monde a tiré un 1.

La valeur de R_N est la longueur de la branche extrême gauche de l'arbre ainsi construit. L'utilisation de la fonction de hachage permet de compter effectivement la cardinalité de l'ensemble (le nombre d'éléments distincts) puisque toutes les occurrences d'un même élément donneront la même clé de hachage.

Variation : Kirshenhofer, Prodinger et Szpankowski [62] ont analysé une version qui prend en compte un phénomène de séparation sous-jacent plus général (mais la méthode de comptage probabiliste qui en résulte ne permet plus de considérer un ensemble avec répétitions).

Remarques :
- L'algorithme Flajolet-Martin peut également être généralisé en considérant la longueur de la branche extrême gauche d'un b-trie
- Des techniques permettent d'obtenir une précision plus importante. Par exemple, on peut exécuter plusieurs fois l'algorithme de comptage mais avec des fonctions de hachage différentes. Dans ce cas, on montre qu'on atteint, en considérant m fonctions de hachage distinctes, en moyenne une précision relative proche de

$$\frac{0,78}{\sqrt{m}}.$$

  Une autre technique pour affiner les résultats est d'utiliser une technique de moyenne stochastique [32]. On ne considère pas m fonctions de hachage, mais l'esprit est le même puisqu'on va considérer $m = 2^B$ chaînes binaires (sur lesquelles on fera une moyenne). On calcule une clé de hachage dont les B premiers bits de cette clé sont utilisés pour sélectionner la chaîne à modifier parmi les $m = 2^B$ chaînes binaires. L'analyse en moyenne conclut à une précision relative identique à celle résultant de la technique précédente (utilisant m fonctions de hachage).

Échantillonage adaptatif. Cette méthode est également basée sur le hachage et a été analysée par P. Flajolet dans [35]. Cela correspond à l'analyse de la branche extrême gauche d'un bucket trie.

À chaque étape, l'algorithme considère une liste contenant au plus m clés de hachage distinctes, où m est un paramètre déterminant la précision de la méthode. On considère également un entier δ qui correspond à la profondeur de l'échantillonage. L'algorithme commence avec une profondeur $\delta = 0$ en construisant une liste sans répétitions des clés de hachage obtenues en parcourant l'ensemble X jusqu'à ce que la liste soit complète (donc jusqu'à la $(m+1)^{\text{e}}$ clé de hachage). À ce moment, on incrémente la profondeur à $\delta = 1$, et on parcourt la liste en supprimant les valeurs ne commençant pas par 0. On continue de parcourir les éléments de X en n'insérant dans la liste que les éléments dont la clé de hachage commence par 0, et ce jusqu'à ce qu'on dépasse à nouveau la capacité m de la pile. On réitère alors le processus en incrémentant la profondeur à $\delta = 2$, et en supprimant de la liste les clés ne commençant pas par 0^δ.
À la fin, lorsqu'on a parcouru tous les éléments de X, on est arrivé à une profondeur δ et la liste contient ℓ clés de hachage. Le cardinal estimé de l'ensemble est donc

$$2^\delta \ell.$$

Le rapport entre l'algorithme et la branche extrême gauche d'un b-trie de capacité $b = m$ est clair. La précision relative pour cette méthode est proche de

$$\frac{1,20}{\sqrt{m}}.$$

Élection d'un chef/perdant

Le problème est le suivant : *comment singulariser quelqu'un dans une assemblée*[3] *?* Afin de résoudre ce problème, on donne à chacun une pièce permettant de jouer à pile ou face.

Le principe de l'élection est le suivant. Chacun, dans une assemblée de N personnes, tire à pile ou face. Ceux qui tirent 0 s'arrêtent de jouer (cela correspond à suivre la branche gauche de l'arbre digital) tandis que ceux qui tirent 1 rejouent (on suit alors la branche droite). Il reste bien sûr à déterminer quand le jeu s'arrête et cela peut donner lieu à plusieurs variantes. Le plus simple est de considérer que le jeu s'arrête lorsqu'il ne reste qu'un joueur. Si à une étape donnée tout le monde tire 1, on recommence l'élection depuis le début.

Il s'agit donc toujours de suivre une branche particulière d'un trie (ici, la branche extrême droite). Ce problème a fait l'objet d'analyses en moyenne assez poussées [82, 49, 59, 30] (nombre moyen de tours et analyse de divers paramètres [82], distribution du nombre de tours [30], modèle biaisé [59]) Les résultats et les méthodes sont typiques de celles utilisées pour l'étude de structures d'arbres digitaux (voir le chapitre 3).

Résolution de conflit pour les communications

Le principe de partitionnement est présent dans le protocole appelé protocole en arbre. On considère un certain nombre de stations qui s'échangent des messages sur un canal partagé [34, 38, 28]. Il s'agit de gérer au mieux les problèmes liés à la collision de messages. Chaque station ne peut distinguer que trois états pour la ligne : 0 si personne n'émet, 1 si une seule station émet, et 2^+ s'il y a collision. Ce protocole fut le premier à être prouvé stable (surpassant donc ainsi le protocole ETHERNET). Le principe du protocole est simple :

[3] Selon les auteurs, l'être singulier est un chef ou un perdant (auquel cas il doit payer une tournée).

> *En cas collision entre un groupe de stations, chaque station effectue un tirage à pile-ou-face ; les stations se répartissent ainsi en 2 groupes et chaque groupe résout, indépendamment de l'autre et récursivement, ses collisions.*

Pour une implantation du protocole, on a recours à une pile.

Analyse d'algorithmes

Les tries en tant que structure digitale apparaît naturellement dans l'analyse de nombreux algorithmes en informatique. Nous proposons deux exemples (non détaillés et très superficiels) où l'analyse s'apparente de manière plus ou moins détournée à l'analyse de paramètres de certains tries.

Factorisation de polynômes. L'algorithme de Lazard est un algorithme de factorisation de polynômes [67] sur un corps fini $\mathbb{F}_p$. Le coût dominant de l'algorithme de Lazard s'apparente pour un polynôme avec r facteurs, à la hauteur d'un trie binaire construit sur r séquences avec les probabilités biaisées $\alpha = \frac{1}{2} - \frac{1}{2p}$ et $\beta = \frac{1}{2} + \frac{1}{2p}$ (voir [44] pour les détails). Asymptotiquement, il est équivalent à

$$\frac{2}{\log(\alpha^2 + \beta^2)} \log r + O(1).$$

Algorithme de simulation. Le problème initial consiste à générer une variable aléatoire X de distribution exponentielle, i.e. telle que

$$\Pr(X \leq x) = 1 - e^{-x},$$

ou de façon équivalente

$$\Pr(x < X \leq x + dx) = e^{-x} dx.$$

Plusieurs algorithmes existent pour générer une telle variable (voir en particulier [63]). La méthode proposée par Von Neumann [79] génère une variable aléatoire par l'intermédiaire du développement de Taylor de sa loi. Cette méthode a été étudiée précisément par Knuth et Yao [64] au niveau le plus bas, c.-à-d. en comptant précisément le nombre de bits à tirer uniformément pour générer le développement binaire (d'une longueur donnée) d'une variable aléatoire de loi exponentielle. Sur 1000 simulations, Knuth et Yao ont conjoncturé que le nombre moyen $\bar{c}(k)$, comptant le nombre de tirages «pile ou face » élémentaires nécessaire pour produire k bits de la variable aléatoire est

$$\bar{c}(k) \approx k + 5{,}4 \pm 0{,}2 + o(1).$$

Cet algorithme de Neumann-Knuth-Yao a été analysé par Flajolet et Saheb dans [42]. L'analyse n'est pas sans rappeler certaines analyses liées aux tries, et effectivement, certains paramètres de l'algorithme ont une définition inductive comparable avec celles qui seront introduites dans le prochain chapitre exposant les méthodes classiques d'analyse de tries.

Chapitre 3

Tries classiques

Sommaire

Comme l'a montré le chapitre précédent, la structure de trie est largement utilisée en informatique dans les applications les plus diverses. Cette structure est d'autant plus séduisante qu'elle se prête également à une étude théorique qui confirme pleinement les bonnes propriétés observées en pratique.

Dans cette thèse, nous nous plaçons dans le cadre de l'analyse en moyenne. Nous commençons par définir ce cadre en prenant comme cas particulier l'analyse en moyenne de paramètres de trie avec le modèle le plus simple et le plus étudié. Puis nous présentons diverses approches pour l'analyse en moyenne de tels tries. Les méthodes générales symboliques, axées autour des séries génératrices et d'équations fonctionnelles ont un grand caractère de généralité et interviendront à plusieurs reprises dans cette thèse (chapitre 6). Elles sont donc exposées plus en détail. Les séries génératrices constituent le cadre d'analyse classique et usuel mais de nouveaux outils sont nécessaires si l'on veut changer de type de source (afin de prendre en compte un certain type de dépendance entre les lettres d'un mot du trie). Des méthodes alternatives exprimant les paramètres de tries en fonction d'alignements exploitent des modèles aléatoires plus complexes comme les chaînes de Markov et sont présentées brièvement. Le chapitre 7 montrera que les outils introduits dans cette thèse (*les opérateurs générateurs*) permettent d'aller plus loin dans cette voie. D'une

manière quasi systématique le dernier maillon dans la chaîne d'analyse de paramètres de trie est la *transformée de Mellin*. Cet outil permet d'extraire d'une expression formelle complexe le comportement asymptotique. La transformée de Mellin intervient plusieurs fois dans cette thèse : au niveau de l'analyse classique des tries dans ce chapitre[1] et également dans une forme plus spécialisée au chapitre 8 pour l'étude des tries généraux.

Le cas des tries classiques pour les paramètres de longueur de cheminement et de taille (ainsi que pour une majoration de la hauteur) est traité en détail (en particulier sur la façon dont on extrait le comportement asymptotique à l'aide de la transformée de Mellin) pour mieux exposer la démarche utilisée.

3.1 Analyse de complexité en moyenne

L'analyse d'un algorithme consiste à déterminer les ressources de calcul nécessaires (le plus souvent temps et espace) à l'exécution de l'algorithme. La taille d'une donnée est en général reliée assez «naturellement» à la place mémoire utilisée. Ainsi la taille d'un tableau sera souvent le nombre d'éléments du tableau. Pour un entier m, on aura recours au nombre de bits nécessaires à sa représentation. Une fois la taille définie, on regroupe les données d'un ensemble I par taille, et l'on considère l'ensemble des données de taille n

$$I_n = \{x \in I \mid |x| = n\},$$

où $|x|$ désigne la taille de x.
On associe alors à un algorithme $\mathcal{A}$ fonctionnant sur cet ensemble de données un paramètre μ défini sur I (en général à valeurs entières) relatif soit à l'exécution même de cet algorithme sur la donnée x (place mémoire, nombre d'opérations fondamentales comme des comparaisons, des affectations...), soit à la configuration de sortie produite (comme le degré du PGCD dans le calcul du PGCD de 2 polynômes de $\mathbb{Z}[X]$ par l'algorithme d'Euclide). Une analyse de complexité cherche à caractériser l'évolution d'un paramètre en fonction de la taille n de la donnée. Si μ_n est la restriction de μ à I_n, on peut définir la complexité dans le meilleur des cas et dans le pire des cas, M_n et P_n par

$$M_n = \inf\{\mu_n(x) \mid x \in I_n\}, \quad P_n = \sup\{\mu_n(x) \mid x \in I_n\}.$$

Ces deux notions sont intéressantes puisqu'elles donnent un encadrement du paramètre μ_n. L'analyse de complexité en moyenne, c.-à-d. l'étude de μ_n sur I_n comme *variable aléatoire*, permet de préciser encore les choses. Il faut définir alors un modèle probabiliste sur I_n précisant la répartition des données. La variable μ_n devient une variable aléatoire. La complexité en moyenne est l'espérance de la variable μ_n :

$$\overline{\mu_n} = \mathrm{E}[\mu_n] = \sum_k k \Pr[\mu_n(X) = k].$$

Le plus souvent, le modèle probabiliste considéré est le modèle uniforme sur I_n. Un exemple classique est celui des tris à base de comparaisons qui considère n nombres réels tirés de façon indépendante selon une loi de probabilité continue sur $[0, 1]$. En raisonnant directement sur l'ordre, cela revient à choisir une permutation de manière uniforme dans $\{1, \ldots, n\}$.

Les diverses notions d'analyse (pire des cas, meilleure des cas et cas en moyenne) sont bien distinctes. Un algorithme très peu efficace sur certaines configurations dans le pire des cas et

[1] Même si dans ce cas, des techniques élémentaires permettent d'arriver au résultat [43].

pratiquement linéaire en moyenne, constitue en fait un algorithme efficace la plupart du temps (donc utilisable en pratique, quitte à laisser de coté les configurations gênantes).

Le cadre de l'analyse en moyenne étant posé, il convient de proposer un modèle pour ce que l'on va appeler un *trie aléatoire*. Le modèle probabiliste choisi pour ce chapitre est d'un intérêt fondamental. Il s'agit d'un modèle avec un alphabet binaire $\mathcal{M} = \{0, 1\}$ et le trie est construit sur un ensemble de mots infinis. Chaque bit de chaque mot est tiré indépendamment dans $\{0, 1\}$ avec des probabilités p et q ($p+q = 1$). Le cas équiprobable $p = q = \frac{1}{2}$ (encore appelé cas symétrique) est le modèle le plus courant dans les analyses et nous nous y attacherons plus particulièrement. On dit encore que les mots sont produits par une source sans mémoire symétrique (voir chapitre 4).

D'autres modèles sont bien sûr possibles. On peut construire par exemple un trie sur n mots de longueur fixée k. Un trie aléatoire est alors construit sur un sous-ensemble de n éléments de $\mathcal{M}^k$. Dans ce modèle, appelé modèle à univers de mots finis, tous les sous-ensembles de mots de cardinal n sont supposés équiprobables avec une probabilité

$$\binom{2^k}{n}/2^k.$$

Une étude détaillée de ce modèle et de ses relations avec le modèle adopté dans ce chapitre (modèle à univers de clés infinies) se trouve dans [41].

3.2 Quelques récurrences

Dans cette partie, on considère un trie construit sur N mots infinis produits par une source binaire sans mémoire symétrique.

Les paramètres étudiés sont la longueur de cheminement L_N (la somme des longueurs des chemins pour aller de la racine à chacune des feuilles) et la taille S_N (le nombre de nœuds internes).

La récurrence se calque sur le principe récursif de construction du trie. Si $N > 1$, le trie T se décompose en deux sous-tries gauche T_0 et droit T_1. Le sous-arbre gauche T_0 regroupe les mots commençant par 0 et le sous-arbre droit les mots commençant par 1. L'arbre T_0 contient k mots et l'arbre T_1 contient $N - k$ mots. La probabilité d'avoir k mots parmi N commençant par 0 est $\binom{N}{k}/2^N$. Il faut encore définir la contribution de la racine dans le paramètre de l'arbre étudié. Celle-ci vaut 1 pour le nombre de nœuds internes et N pour la longueur de cheminement. Cette décomposition d'un paramètre selon sa contribution à la racine et dans les deux sous-arbres est à la base de la plupart des analyses de tries. Ce principe sera à nouveau utilisé pour l'analyse de tries généraux avec des sources générales au chapitre 8.

Les valeurs moyennes L_N et S_N de la longueur de cheminement et de la taille d'un trie construit sur N mots binaires aléatoires vérifient alors

$$L_N = N + \frac{1}{2^N}\sum_{k=0}^{N}\binom{N}{k}(L_k + L_{N-k}) \qquad \text{pour } N > 1 \text{ avec } L_0 = L_1 = 0, \tag{3.1}$$

$$S_N = 1 + \frac{1}{2^N}\sum_{k=0}^{N}\binom{N}{k}(S_k + S_{N-k}) \qquad \text{pour } N > 1 \text{ avec } S_0 = S_1 = 0. \tag{3.2}$$

Ces récurrences sont utilisables sous cette forme mais l'utilisation de l'outil puissant que sont les séries génératrices permet de simplifier grandement la résolution de telles équations.

Remarque. On peut récrire la récurrence pour $\frac{L_N}{N}$ (pour $N > 1$), on obtient

$$\frac{L_N}{N} = 1 + \frac{2}{2^N}\sum_{k=0}^{N}\frac{\binom{N}{k}}{N}L_k = 1 + \frac{2}{2^N}\sum_{k=0}^{N-1}\binom{N-1}{k}\frac{L_k}{N-1}.$$

On peut donc se concentrer sur un seul des paramètres, car clairement $L_N = NS_{N-1}$.

On peut également établir des récurrences relatives à la hauteur d'un trie aléatoire. Soit $p_{N,k}$ la probabilité qu'un trie T construit sur N mots ait une hauteur $H(T)$ inférieure ou égale à k. Pour un trie T de sous-tries gauche et droit T_0 et T_1, l'événement $\{H(T) \leq k\}$ s'écrit $\{H(T_0) \leq k-1$ et $H(T_1) \leq k-1\}$. La conjonction de deux événements indépendants se traduit par le produit. On obtient

$$p_{N,k} = \frac{1}{2^n}\sum_{i=0}^{N} p_{i,k-1}p_{N-i,k}.$$

3.3 Séries génératrices

Dans cette section, nous introduisons un des concepts centraux pour l'analyse en moyenne d'algorithmes. Nous commençons par une présentation succincte des séries génératrices ordinaires et exponentielles et de leurs applications immédiates.
Les séries génératrices sont très intéressantes à plus d'un titre. Elles permettent par exemple de résoudre les récurrences (3.1) et (3.2). De plus, les méthodes symboliques permettent de manipuler en parallèle les objets étudiés et les séries génératrices associées. Les séries génératrices sont donc plus qu'un outil. Elles offrent un moyen de pénétrer au cœur des structures étudiées.
Parfois la résolution de récurrences revient à résoudre une équation fonctionnelle mettant en jeu des séries génératrices (on peut d'ailleurs faire un parallèle intéressant avec les méthodes du chapitre 7). La connaissance de cette équation fonctionnelle fournit de nombreux renseignements sur sa solution même si une solution explicite n'est pas connue.

3.3.1 Présentation

Cette section présente l'outil classique des séries génératrices et leur utilisation pour résoudre des récurrences.

Séries génératrices ordinaires et exponentielles

DÉFINITION 3.1 (SÉRIE GÉNÉRATRICE ORDINAIRE). *La série génératrice ordinaire (*SGO*) associée à une suite $(a_i)_{i\geq 0}$ est la série formelle*

$$A(z) = \sum_{k=0}^{\infty} a_k z^k.$$

L'exemple le plus simple de série génératrice ordinaire correspond à la suite constante égale à 1 et vaut $1 + z + z^2 + z^3 + \cdots = \frac{1}{1-z}$.

DÉFINITION 3.2 (SÉRIE GÉNÉRATRICE EXPONENTIELLE). *La série génératrice exponentielle (*SGE*) associée à une suite $a_0, a_1, \ldots, a_k, \ldots$ est la fonction*

$$\widehat{A}(z) = \sum_{k=0}^{\infty} a_k \frac{z^k}{n^k}.$$

La série génératrice associée à la même suite constante $1, 1, 1, \ldots$ est donc $1 + \frac{z}{1!} + \frac{z^2}{2!} + \cdots = \mathrm{e}^z$. Les sommes considérées ici peuvent ou non converger, mais nous ne nous intéresserons que très peu, dans un premier temps, à ces problèmes de convergence et ce, pour deux raisons. La première réside dans le fait que les manipulations que nous opérons sur les séries génératrices sont parfaitement définies si nous les considérons en tant que séries formelles, même en l'absence de convergence. Ensuite, la nature même des suites étudiées en analyse d'algorithme assure la convergence des séries, au moins pour z suffisamment petit. Dans beaucoup d'applications en analyse d'algorithme, l'analyse se divise typiquement en deux parties. Les relations formelles pour la série entière permettent d'obtenir des formules explicites pour les séries génératrices. Puis, on étudie les propriétés analytiques des séries génératrices en détail (ici la convergence des séries joue un grand rôle) afin de revenir aux paramètres étudiés et pour obtenir des formules explicites.

Séries génératrices comme solutions de récurrences

Les séries génératrices sont un bon moyen de résoudre des récurrences. Si l'on examine les équations (3.1) et (3.2), en multipliant par z^N les deux membres de chaque équation, on obtient une expression sous forme d'un produit de convolution qui incite (fortement !) à utiliser les séries génératrices exponentielles. En effet, pour deux séries génératrices exponentielles

$$\widehat{A}(z) = \sum a_k \frac{z^k}{k!} \text{ et } \widehat{B}(z) = \sum b_k \frac{z^k}{k!}$$

le terme général c_k de la série produit $\widehat{C}(z) = \widehat{A}(z)\widehat{B}(z)$ est

$$c_k = \sum_{i=0}^{k} \frac{a_i}{i!} \frac{b_{k-i}}{(k-i)!}.$$

On aboutit donc à une équation fonctionnelle pour la série génératrice exponentielle

$$\widehat{L}(z) = z\mathrm{e}^z - z + 2\mathrm{e}^{z/2}\widehat{L}(z/2). \tag{3.3}$$

En «déroulant» cette relation, on obtient

$$\begin{aligned}
\widehat{L}(z) &= z\mathrm{e}^z - z + 2\mathrm{e}^{z/2}\widehat{L}(z/2) \\
&= z\mathrm{e}^z - z + 2\mathrm{e}^{z/2}\left(\frac{z}{2}\mathrm{e}^{z/2} - \frac{z}{2} + 2\mathrm{e}^{z/4}\widehat{L}(z/4)\right) \\
&= z(e^z - 1) + z(e^z - e^{z/2}) + 4e^{3z/4}\widehat{L}(z/4) \\
&= z(e^z - 1) + z(e^z - e^{z/2}) + z(e^z - e^{3z/4}) + 8e^{7z/8}\widehat{L}(z/8) \\
&\vdots \\
&= z\sum_{j\geq 0}\left(e^z - e^{(1-2^{-j})z}\right).
\end{aligned} \tag{3.4}$$

Il reste à extraire le coefficient L_N à partir de cette équation. On a

$$L_N = N!\,[z^N]\,\widehat{L}(z) = N\sum_{j\geq 0}\left(1 - \left(1 - \frac{1}{2^j}\right)^{N-1}\right).$$

(Nous montrerons par la suite un autre moyen d'obtenir une telle expression.) Une telle somme est typique de celles rencontrées pour les tries binaires symétriques. Le comportement asymptotique pour $N \to \infty$ peut être étudié par des techniques élémentaires ou en utilisant des techniques plus élaborées comme la transformée de Mellin (voir la section 3.8).

3.3.2 Utilisation pour l'analyse en moyenne

Nous venons de voir une application pour l'analyse d'algorithmes puisque la longueur de cheminement externe n'est autre que le nombre de bits examinés par l'algorithme de tri *radix*. D'autres séries génératrices sont très importantes pour l'analyse en moyenne, comme les séries génératrices de probabilités et les séries génératrices bivariées.

Série génératrice de probabilité. Les séries génératrices de probabilités permettent de manipuler des probabilités et de simplifier le calcul des valeurs moyennes et des moments.

DÉFINITION 3.3. *Soit une variable aléatoire X prenant des valeurs entières et positives, avec $p_k = \Pr\{X = k\}$, la fonction $P(u) = \sum_{k\geq 0} p_k u^k$ est appelée la série génératrice des probabilités (*SGP*) de la variable aléatoire.*

Grâce au théorème suivant, on est en mesure de trouver les valeurs moyennes et les variances (et plus généralement tout moment) sans passer par des calculs impliquant des sommes discrètes.

THÉORÈME 3.4 (VALEUR MOYENNE ET VARIANCE À PARTIR DES SGP). *Soit $P(u)$ la série génératrice de probabilités d'une variable aléatoire X. La valeur moyenne de X est égale à $P'(1)$ et la variance est égale à $P''(1) + P'(1) - P'(1)^2$.*

Preuve. La preuve de ce théorème classique est reproduite ici car elle introduit des calculs simples et éclairant sur les séries génératrices. Si $p_k = \Pr\{X = k\}$, alors

$$P'(1) = \sum_{k\geq 0} k p_k \; u^{k-1}\big|_{u=1} = \sum_{k\geq 1} k p_k,$$

la valeur moyenne de X, par définition. De façon similaire, en remarquant que $P(1) = \sum p_k = 1$, le résultat pour la variance s'ensuit directement

$$\begin{aligned}
\mathrm{Var}(X) &= \mathrm{E}[(X - E[X])^2] \\
&= \sum_{k\geq 0} (k - P'(1))^2 p_k \\
&= \sum_{k\geq 0} k^2 p_k - 2\sum_{k\geq 0} k P'(1) p_k + \sum_{k\geq 0} P'(1)^2 p_k \\
&= \sum_{k\geq 0} k^2 p_k - P'(1)^2 \\
&= P''(1) + P'(1) - P'(1)^2.
\end{aligned}$$

□

De manière générale, la quantité $\mathrm{E}[X^r] = \sum_k k^r p_k$ est le r^{e} moment et s'exprime en fonction des dérivées successives de $P(u)$ en $u = 1$.

Série génératrice bivariée. L'analyse en moyenne d'algorithmes s'intéresse à non seulement compter des structures d'une certaine taille, mais aussi, conjointement, à étudier les valeurs de divers paramètres (par exemple le nombre de mots d'un trie conjointement avec la hauteur). Nous utilisons à cette fin les séries génératrices *bivariées*. Ce sont des séries formelles de deux variables relatives à des suites doubles. Un indice se rapporte à la taille du problème, l'autre se réfère au paramètre analysé.

DÉFINITION 3.5 (SÉRIE GÉNÉRATRICE BIVARIÉE). *Soit $(a_{n,k})_{n,k\geq 0}$ une suite double. La série*

$$A(z,u) = \sum_{n\geq 0}\sum_{k\geq 0} a_{n,k}u^k z^n$$

*est appelée la série génératrice bivariée (*SGB*) de la suite. Nous utilisons la notation $[z^n u^k]A(z,u)$ pour désigner $a_{n,k}$, $[z^n]A(z,u)$ pour désigner $\sum_{k\geq 0} a_{n,k}u^k$ et $[u^k]A(z,u)$ pour désigner $\sum_{n\geq 0} a_{n,k}z^n$.*

Bien entendu, cette définition est relative aux séries génératrices bivariées *ordinaires* et peut être rendue «exponentielle» en divisant chaque terme par $n!$. Ainsi la SGB exponentielle de $(a_{n,k})$ est

$$\widehat{A}(z,u) = \sum_{n\geq 0}\sum_{k\geq 0} a_{n,k}u^k \frac{z^n}{n!}.$$

Nous utilisons les SGB dans le cadre suivant. Pour un élément α de $\mathcal{P}$, où $\mathcal{P}$ est une classe de structures combinatoires, on considère la fonction de coût $c(\alpha)$ donnant la valeur d'un paramètre sur l'entrée α. Alors la série génératrice bivariée pour cette fonction de coût est

$$P(z,u) = \sum_{\alpha\in\mathcal{P}} u^{c(\alpha)}z^{|\alpha|} = \sum_{n\geq 0}\sum_{k\geq 0} p_{n,k}u^k z^n$$

où $p_{n,k}$ est le nombre de structures de taille n et de coût k.
On peut encore écrire $P(z,u)$ sous la forme

$$P(z,u) = \sum_{n\geq 0} p_n(u)z^n \text{ avec } p_n(u) = \sum_{k\geq 0} p_{n,k}u^k,$$

pour grouper selon la taille. De manière symétrique l'écriture

$$P(z,u) = \sum_{k\geq 0} q_k(z)u^k \text{ avec } q_k(z) = \sum_{n\geq 0} p_{n,k}z^n,$$

permet de grouper les termes de la série relatifs à un même coût.
On voit que $P(z,1)$ est la série génératrice énumérative de la classe $\mathcal{P}$. En effet le terme $[z^n]\,P(z,1)$ est le nombre d'objets de taille n.

DÉFINITION 3.6 (SÉRIE GÉNÉRATRICE CUMULÉE). *Soit $\mathcal{P}$ une classe de structures combinatoires avec une série génératrice bivariée $P(z,u)$ (associée à une fonction de coût c). Alors la fonction*

$$\frac{\partial}{\partial u}P(z,u)\bigg|_{u=1} = \sum_{\alpha\in\mathcal{P}} c(\alpha)z^{|\alpha|}$$

est la série génératrice cumulée. Si $\mathcal{P}_n$ est la classe de toutes les structures de $\mathcal{P}$ de taille n, alors la somme

$$\sum_{\alpha\in\mathcal{P}_n} c(\alpha)$$

est définie comme le coût cumulé pour les structures de taille n.

Cette terminologie s'explique par le fait que le coefficient de z^n dans la série génératrice cumulée est le coût cumulé.

Le théorème suivant permet de calculer la valeur moyenne de la fonction de coût lorsque les structures d'une taille donnée sont équiprobables.

THÉORÈME 3.7 (SÉRIE GÉNÉRATRICE BIVARIÉE ET COÛT MOYEN). *Pour une série génératrice bivariée $P(z,u)$ pour une classe de structures combinatoires, le coût moyen pour toutes les structures de taille n sous un modèle uniforme est*

$$\frac{[z^n]\frac{\partial}{\partial u}\,P(z,u)|_{u=1}}{[z^n]P(z,1)}.$$

Preuve. Le rapport du coût cumulé sur le nombre de structures de taille n correspond bien à la valeur moyenne recherchée. □

Nous nous servirons de ces séries génératrices cumulées dans le chapitre 6.

3.4 Méthodes symboliques sur les tries

À partir d'une récurrence relative à un paramètre structurel du trie, on peut calculer la série génératrice correspondante, puis une équation fonctionnelle comme dans (3.3). On peut en fait bien souvent se passer de l'étape établissant la récurrence. L'idée consiste à relier directement sous un modèle probabiliste donné les paramètres structurels de trie à des équations fonctionnelles de séries génératrices. C'est précisément la méthode symbolique exposée dans [41] (voir également [43]) qui relie de façon algébrique les paramètres et les séries génératrices.
L'analyse consiste à exprimer les paramètres sous une forme adaptée puis à en déduire une équation fonctionnelle. Celle-ci peut le plus souvent être «déroulée» comme dans (3.3) pour obtenir une expression exacte. Dans le cas des tries, le comportement asymptotique est ensuite souvent établi à l'aide de la transformée de Mellin qui constitue le dernier maillon de la chaîne d'analyse.

Pour un ensemble X de n mots binaires, nous utiliserons la notation $\underline{\mathrm{T}}_{[0]}(X)$ et $\underline{\mathrm{T}}_{[1]}(X)$ pour désigner les mots obtenus à partir de ceux de X grâce au décalage $\underline{\mathrm{T}}$ et regroupant pour le premier les mots commençant par 0 et pour le deuxième les mots commençant par 1.

$$\underline{\mathrm{T}}_{[b]}(X) = \{\underline{\mathrm{T}}(x) | x \in X \text{ et } \underline{\sigma}(x) = b\}, \quad (\text{pour } b \in \mathcal{M} = \{0,1\}).$$

Dans ce paragraphe, un trie construit sur n mots produits indépendamment par une source sans mémoire binaire symétrique. Avec nos notations la probabilité que l'ensemble X de cardinal n se partage en deux sous-ensembles $\underline{\mathrm{T}}_{[0]}(X)$ et $\underline{\mathrm{T}}_{[1]}(X)$ de cardinaux k et $n-k$ est

$$\frac{1}{2^n}\binom{n}{k}.$$

Très souvent, on peut exprimer un paramètre v récursivement par rapport aux sous-tries en tenant compte de la contribution w à la racine

$$v(X) = v(\underline{\mathrm{T}}_{[0]}(X)) + v(\underline{\mathrm{T}}_{[1]}(X)) + w(X). \tag{3.5}$$

Le paramètre est défini de façon inductive grâce à des opérations «élémentaires» et aux opérateurs $\underline{\mathrm{T}}_{[0]}$, $\underline{\mathrm{T}}_{[1]}$. On associe à un paramètre a la série génératrice

$$\widehat{a}(z) = \sum_X \frac{a(X)}{|X|!} z^{|X|}.$$

Le lemme suivant (tiré de [41]) permet de traduire les relations sur les paramètres sur les séries génératrices.

LEMME 3.8 (TRADUCTION DES PARAMÈTRES). *Si les paramètres sur les tries satisfont les relations*

$$a(X) = \lambda b(X), \qquad a(X) = b(X) + c(X), \qquad a(X) = b(\underline{\mathrm{T}}_{[0]}(X))c(\underline{\mathrm{T}}_{[1]}(X)),$$

alors les séries génératrices exponentielles vérifient

$$\widehat{a}(z) = \lambda\widehat{b}(z), \qquad \widehat{a}(z) = \widehat{b}(z) + \widehat{c}(z), \qquad \widehat{a}(z) = \widehat{b}(z/2)\widehat{c}(z/2).$$

Preuve. Les deux premières relations sont triviales. La forme produit provient du fait que si X est un ensemble aléatoire de n mots de $\mathcal{M}^{\mathbb{N}}$, l'ensemble $\underline{\mathrm{T}}_{[0]}(X)$, conditionné à être de cardinal k, est un ensemble aléatoire de k mots de $\mathcal{M}^{\mathbb{N}}$. (On utilise ici fortement le fait que les symboles constituant un mot sont indépendants.) La dernière relation est prouvée par

$$\begin{aligned}\widehat{a}(z) = \sum_{n\geq 0} a_n \frac{z^n}{n!} &= \sum_{n\geq 0}\sum_{k=0}^{n} \frac{1}{2^n}\binom{n}{k} b_k c_{n-k} z^n \\ &= \left(\sum_{n\geq 0} b_n \frac{(z/2)^n}{n!}\right)\left(\sum_{n\geq 0} c_n \frac{(z/2)^n}{n!}\right) \\ &= \widehat{b}(z/2)\widehat{c}(z/2).\end{aligned}$$

□

On peut également définir des «briques» de base pour exprimer les paramètres de trie.

LEMME 3.9 (PARAMÈTRES PARTICULIERS). *Les valuations :*

$$a(X) = 1, \qquad b(X) = \delta_{|X|,p}, \qquad c(X) = |X|,$$

(où $\delta_{n,p}$ désigne le symbole de Kronecker[2] *admettent pour séries génératrices exponentielles*

$$\widehat{a}(z) = e^z, \qquad \widehat{b}(z) = \frac{z^p}{p!}, \qquad \widehat{c}(z) = ze^z.$$

Les deux lemmes précédents permettent de traduire la relation (3.5). On obtient l'expression

$$\widehat{v}(z) = 2e^{z/2}\widehat{v}(z/2) + w(z).$$

Avant d'en venir à des exemples d'applications, remarquons que cette équation peut être résolue par itération à l'aide du lemme suivant.

LEMME 3.10 (LEMME ITÉRATIF). *On suppose que la* SGE *vérifie*

$$\widehat{f}(z) = ce^{z/2}\widehat{f}(z/2) + \widehat{w}(z).$$

et qu'il existe un entier d pour lequel les conditions suivantes sont vérifiées :

[2] $\delta_{n,p} = 1$ si $n = p$ et 0 sinon.

– *la fonction $\widehat{w}(z)$ vérifie une condition de contraction*

$$\widehat{w}(z) = O(z^d) \text{ lorsque } z \to 0,$$

– *la fonction f vérifie la condition initiale $f(0) = f'(0) = f''(0) = \cdots = f^{(d-1)} = 0$.*
Alors l'équation admet une unique solution de la forme

$$\widehat{f}(z) = \sum_{j\geq 0} c^j \ \widehat{w}(z/2^j) e^{z(1-\frac{1}{2^j})},$$

et les coefficients f_n sont donnés par la formule

$$f_n = \sum_{k=0}^{n} \binom{n}{k} \frac{w_k^*}{1-2^{-k}}, \text{ où } w_k^* = k![z^k]e^{-z}\widehat{w}(z). \tag{3.6}$$

Preuve. Les conditions initiales ainsi que la condition de contraction assurent la convergence de la somme infinie. Pour la deuxième partie, on considère $\tilde{f}(z) = e^{-z}\widehat{f}(z)$ (on verra au chapitre 8 que cela équivaut à travailler dans le modèle de Poisson correspondant). La fonction $\tilde{f}(z)$ vérifie

$$\tilde{f}(z) = c\tilde{f}(z/2) + \tilde{w}(z) \qquad \text{(avec } \tilde{w}(z) = e^{-z}\widehat{w}(z)\text{)}.$$

En identifiant les coefficients, on obtient la relation $\tilde{f}_n = c2^{-n}\tilde{f}_n + \tilde{w}_n$. La relation (3.6) s'ensuit alors puisque les coefficients de $f(z)$ s'obtiennent par convolution à partir de ceux de $\tilde{f}(z)$ et e^z. □

3.5 Analyse

On peut à l'aide de ces trois lemmes analyser un grand nombre de paramètres liés aux tries.

Longueur de cheminement et taille. Les paramètres s du nombre de nœuds et ℓ de longueur de cheminement s'expriment ainsi grâce aux paramètres élémentaires. En effet on a

$$s(X) = 1 - \delta_{|X|,0} - \delta_{|X|,1} + s(\underline{\mathrm{T}}_{[0]}) + s(\underline{\mathrm{T}}_{[1]}$$
$$\ell(X) = |X| - \delta_{|X|,1} + \ell(\underline{\mathrm{T}}_{[0]}) + \ell(\underline{\mathrm{T}}_{[1]}\,.$$

La première relation exprime le fait que la taille d'un trie construit sur X est la somme des tailles des sous-arbres gauche et droit plus un (contribution de la racine) si X contient au moins deux éléments. La longueur de cheminement, toujours si le nœud existe (i.e. $|X| > 1$), est égale à la somme des longueurs de cheminement des sous-arbres gauche et droit plus la contribution de la racine $|X|$. En appliquant les lemmes 3.8 et 3.9, on obtient directement les équations fonctionnelles

$$\widehat{s}(z) = e^z - 1 - z + 2e^{z/2}\,\widehat{s}(z/2)$$
$$\widehat{\ell}(z) = ze^z - z + |X| + 2e^{z/2}\,\widehat{\ell}(z/2).$$

On calcule les solutions exactes sous forme de sommes infinies grâce au lemme 3.10.

$$\widehat{s}(z) = \sum_{k\geq 0} 2^k \left(e^z - e^{z(1-\frac{1}{2^k})} - \frac{z^k}{2^k} e^{z(1-\frac{1}{2^k})} \right) \tag{3.7}$$

$$\widehat{\ell}(z) = \sum_{k\geq 0} z(e^z - e^{z(1-\frac{1}{2^k})}). \tag{3.8}$$

L'équation (3.8) est la solution que l'on avait obtenue en (3.4) à partir de la récurrence. Contrairement à la preuve calculatoire précédente, on obtient directement la série génératrice à partir de la définition inductive du paramètre. On trouve également les expressions des coefficients s_n et ℓ_n

$$s_n = \sum_{k\geq 0} 2^k (1 - (1 - \frac{1}{2^k})^n - \frac{n}{2^k}(1 - \frac{1}{2^k})^{n-1}) \tag{3.9}$$

$$\ell_n = n \sum_{k\geq 0} (1 - (1 - \frac{1}{2^k})^{n-1}). \tag{3.10}$$

On obtiendra Ces expressions autrement au chapitre 4.

Hauteur. La quantité $p_k(X)$ vaut 1 si le trie est de hauteur inférieure à k et 0 sinon. Elle obéit à la définition inductive

$$p_k(X) = p_{k-1}(\underline{\mathrm{T}}_{[0]}(X)) p_{k-1}(\underline{\mathrm{T}}_{[1]}(X)).$$

De plus, la valeur moyenne $p_{k,n}$ pour les ensembles X de cardinal n est exactement la probabilité qu'un ensemble de n mots ait un trie associé de hauteur au plus k. On trouve l'équation fonctionnelle grâce aux lemmes 3.8 et 3.9

$$\widehat{p}_k(z) = \widehat{p}(z/2)^2 = \widehat{p}_0(z/2^k)^{2^k} = (1 + z/2^k)^{2^k}.$$

L'espérance de la hauteur d'un trie construit sur n mots est exactement

$$h_n = \sum_{k\geq 0} (1 - p_{k,n}) = \sum_{k\geq 0} \left[1 - n![z^n](1 + z/2^k)^{2^k}\right]. \tag{3.11}$$

Remarques.

1. Ces méthodes très puissantes permettent également d'obtenir les expressions exactes de la valeur moyenne d'autres paramètres (comme la longueur de cheminement d'un patricia trie, la recherche dans un k-d trie, voir [41]).
2. Ces méthodes s'adaptent à un modèle probabiliste où les probabilités p et $q = 1 - p$ d'avoir 0 ou 1 ne valent plus $\frac{1}{2}$ (cas biaisé). Les lemmes 3.8 et 3.10 s'adaptent. Les propriétés additives sont toujours vraies. La principale différence se situe au niveau des paramètres multiplicatifs, pour lesquels la relation

   $$v(X) = w(\underline{\mathrm{T}}_{[0]}(X)) t(\underline{\mathrm{T}}_{[1]}(X)),$$

   se traduit par

   $$\widehat{v}(z) = \widehat{w}(pz)\widehat{t}(qz).$$

3. Ces lemmes peuvent également être étendus à un alphabet non binaire [41].

Le point commun à toutes ces méthodes est de ne considérer que des sources sans mémoire pour les mots produits. Cette thèse examinera un modèle de source très général (qui englobe en particulier les sources sans mémoire mais aussi les chaînes de Markov) qui permet de prendre en compte des dépendances entre les symboles (voir chapitre 4). Ce modèle qui englobe la majeure partie des modèles classiques envisagés pour les tries utilisera un autre type d'approche et établira des résultats très généraux sur les tries.

3.6 Séries génératrices bivariées

Les séries génératrices étudiées dans la section précédente étaient univariées. Elles correspondentà des séries génératrices d'espérances pour la taille et la longueur de cheminement et à une série génératrice de probabilités pour la hauteur. Les séries génératrices bivariées ont un pouvoir d'expression plus grand puisqu'elles prennent en compte deux paramètres conjointement.

Nous nous intéressons par exemple au paramètre donnant la profondeur à laquelle la clé est insérée. Les probabilités $p_{n,k}$ qu'une clé soit insérée à une profondeur k dans un trie contenant $n+1$ mots donnent lieu à une série génératrice exponentielle de probabilités [53]. Le modèle aléatoire pour les mots peut être ici biaisé (de paramètres p et $q = 1-p$). Toute l'information sur ces probabilités est résumée dans la série génératrice

$$\widehat{P}(z,u) = u[p\widehat{P}(pz,u) + q\widehat{P}(qz,u)] + (1-u)e^{-z}. \tag{3.12}$$

Remarque. Une telle équation se prête naturellement à l'analyse grâce à la transformée de Mellin présentée à la section 3.8.

On peut également trouver des relations pour la série génératrice bivariée de la hauteur. Considérons la série génératrice exponentielle bivariée $\widehat{H}(z,u)$ associée à la hauteur. Le calcul se passe en deux temps. D'abord on va étudier la série génératrice exponentielle des probabilités $p_{k,n}$ qu'un trie construit sur n mots soit de hauteur inférieure ou égale à k. On a déjà trouvé dans le cas symétrique $p = q = \frac{1}{2}$ la série génératrice

$$\widehat{p}_k(z) = \sum_{k\geq 0} p_{k,n}\, z^n/n! = (1+z/2^k)^{2^k}.$$

Une formule existe également dans le cas biaisé. On peut utiliser la récurrence

$$\begin{cases} p_{k,n} = \sum_{n_1+n_2=n} \binom{n}{n_1,n_2} p^{n_1}q^{n_2}p_{k-1,n_1}p_{k-1,n_2}, & k\geq 1, n\geq 0 \\ p_{0,n} = 0 & n\leq 1 \\ p_{0,n} = 1 & n>1, \end{cases}$$

ou encore appliquer une version biaisée des lemmes 3.8 et 3.10. On obtient

$$\widehat{p}_k(z) = \sum_{n\geq 0} p_{k,n}\frac{z^n}{n!} = \prod_{k_1+k_2=k} \left(e^{-zp^{k_1}q^{k_2}}(1+zp^{k_1}q^{k_2})\right)^{\binom{k}{k_1,k_2}}.$$

On utilise le fait que la probabilité qu'un trie construit sur n mots soit de hauteur k est $(p_{k,n} - p_{k-1,n})$ si $k>0$ et $p_{0,n}$ sinon. La série génératrice de la hauteur s'écrit alors

$$\begin{aligned} \widehat{h}(z,u) &= \sum_{n\geq 0} h_n(u)\frac{z^n}{n!} \\ &= \sum_{n\geq 0}\frac{z^n}{n!}\sum_{k\geq 0}(p_{k,n}-p_{k-1,n})u^k + p_{0,n} \\ &= (1-u)\sum_{k\geq 0} u^k p_{k,n}\frac{z^n}{n!} = (1-u)\sum_{k\geq 0} p_k(z)u^k. \end{aligned}$$

Cette équation relie la série génératrice exponentielle bivariée de la hauteur $\widehat{h}(z,u)$ à la série génératrice exponentielle $\widehat{p}_k(z)$ des probabilités qu'un trie soit de hauteur inférieure à k.

Le fait de trouver une équation fonctionnelle ne constitue que le premier pas de l'analyse en moyenne. Plusieurs cas se présentent alors : on a une solution explicite sur laquelle toute la machinerie des séries génératrices est applicable (extraction de coefficients, calculs des moments) ; il peut également arriver qu'on ne conaisse pas de solution explicite mais la structure de l'équation fonctionnelle donne des informations notamment sur le comportement asymptotique du paramètre (l'outil employé est alors l'analyse complexe).

3.7 Utilisation des alignements

Les méthodes des paragraphes précédents permettent de prendre en compte seulement les modèles indépendants où les mots sont produits par une source sans mémoire (cf. chapitre 4). Afin de considérer d'autres modèles, plusieurs auteurs ont proposé des expressions alternatives des paramètres de tries qui font appel à la notion d'alignement [54, 96, 4, 60]. On peut prendre en compte les dépendances de type markovienne (la probabilité d'avoir un symbole dépend du symbole précédent) ou encore les dépendances survenant en examinant les suffixes successifs d'un texte.

Considérons une suite $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ de n mots sur un alphabet fini $\mathcal{M}$ de taille r. On définit la matrice des alignements $(C_{i,j})_{1\leq i,j\leq n}$. Le terme $C_{i,j}$ (pour $i \neq j$) désigne la longueur du plus long préfixe commun aux deux mots $\boldsymbol{x}_i$ et $\boldsymbol{x}_j$. Ainsi, $C_{i,j} = k$ si et seulement si $\boldsymbol{x}_i$ et $\boldsymbol{x}_j$ coïncident exactement sur les k premiers caractères mais diffèrent sur le $(k+1)^{\text{e}}$. La matrice $(C_{i,j})$ est symétrique, i.e. $C_{i,j} = C_{j,i}$.

Exemple. Soit $X = (aabba\ldots, ababb\ldots, abbaa\ldots, babaa\ldots)$ alors la matrices des alignements est

$$C = \begin{pmatrix} \infty & 1 & 1 & 0 \\ 1 & \infty & 2 & 0 \\ 1 & 2 & \infty & 0 \\ 0 & 0 & 0 & \infty \end{pmatrix}.$$

La plupart des paramètres de trie ont une interprétation en termes d'alignements [96]. Par exemple, la profondeur $D_n^{(i)}$ du nœud externe correspondant à $\boldsymbol{x}_i$ s'exprime comme la longueur maximale parmi les alignements (pour parvenir au dernier nœud interne) plus un pour le lien reliant ce nœud interne au nœud externe

$$D_n^{(i)} = 1 + \max_{1\leq i<j\leq n}\{C_{i,j}\}.$$

La profondeur moyenne d'une feuille D_n, la longueur de cheminement L_n, la hauteur H_n s'expriment à l'aide de $D_n^{(i)}$ et donc des alignements $C_{i,j}$. En effet on a

$$\begin{aligned} D_n &= \frac{1}{n}\sum_{i=1}^{n} D_n^{(i)} = 1 + \frac{1}{n}\sum_{i=1}^{n} \max_{j\neq i}\{C_{i,j}\} \\ L_n &= \sum_{i=1}^{n} D_n^{(i)} = n + \sum_{i=1}^{n} \max_{j\neq i}\{C_{i,j}\} \\ H_n &= \max_{1\leq i\leq n}\{D_n^{(i)}\} = 1 + \max_{1\leq i<j\leq n}\{C_{i,j}\}. \end{aligned}$$

Sur l'exemple précédent, on obtient $D_n = 9/4$, $L_n = 9$, $H_n = 3$.

Suivant la démarche de Jacquet et Szpankowski [54], nous allons nous concentrer sur le paramètre D_n.

Les mots sont indépendants les uns des autres donc $D_n^{(i)}$ et D_n ont même distribution pour tout i. la définition de D_n se réduit donc à

$$D_n = 1 + \max\{C_{1,1}, \ldots, C_{1,n}\}.$$

En appliquant le principe d'inclusion-exclusion, on obtient

$$\Pr\{D_n > k\} = \Pr\{\bigcup_{j=2}^{n}[C_{1,j} > k]\} = \frac{1}{n}\sum_{j=2} n(-1)^j \binom{n}{j} j \Pr\{C_{1,2} > k, \ldots, C_{1,j} > k\}. \tag{3.13}$$

Cette formule est vraie dans un cadre très général et reste valable pour une distribution quelconque des symboles dans chaque mot. La série génératrice ordinaire des probabilités associée au paramètre D_n s'écrit donc

$$\begin{aligned} P_n(z) &= \sum_{k\geq 0} \Pr\{D_n = k\} z^k \\ &= \frac{1}{n}\sum_{j=2} n(-1)^j \binom{n}{j} j \\ &\qquad [\Pr\{C_{1,2} > k-1, \ldots, C_{1,j} > k-1\} - \Pr\{C_{1,2} > k, \ldots, C_{1,j} > k\}] z^k. \end{aligned} \tag{3.14}$$

Considérons maintenant le modèle le plus simple avec un alphabet binaire $\mathcal{M} = \{0, 1\}$ où les symboles sont émis indépendamment avec les probabilités p et q. Alors la probabilité $\Pr\{C_{1,2} > k, \ldots, C_{1,j} > k\}$ s'écrit

$$\Pr\{C_{1,2} > k, \ldots, C_{1,j} > k\} = (p^j + q^j)^{k+1},$$

et la série génératrice $P_n(z)$ de l'équation (3.14) devient après des calculs simples

$$P_n(z) = 1 - \frac{1-z}{n}\sum_{j=2}^{n}(-1)^j \binom{n}{j} \frac{j(p^j + q^j)}{1 - z(p^j + q^j)}. \tag{3.15}$$

Ainsi une application du théorème 3.4 donne une expression de l'espérance de D_n

$$\mathrm{E}[D_n] = \frac{1}{n}\sum_{j=2}^{n}(-1)^j \binom{n}{j} \frac{j(p^j + q^j)}{1 - (p^j + q^j)}. \tag{3.16}$$

Le comportement asymptotique de telles sommes alternées (3.15) et (3.16) est obtenu grâce à une analyse de Mellin [94] (bien que l'utilisation en soit légèrement différente que celle qui en est faite habituellement dans les tries).

On peut considérer un modèle lié aux chaînes de Markov (cf. chapitre 4) qui prend en compte une dépendance limitée entre symboles successifs (mais toujours en supposant que les mots sont indépendants) et c'est alors la même méthode qui s'applique. Le cas où les mots sont les suffixes d'un même texte (et ne sont donc plus indépendants) est abordé dans [4]. Cela est en rapport avec l'arbre des suffixes du chapitre 2.

Cette approche se prête également assez bien à une vision dans le cadre des intervalles fondamentaux (chapitre 4). Cela constitue d'ailleurs un axe de recherche dans le prolongement de cette thèse.

3.8 Transformée de Mellin

La transformée de Mellin est omniprésente pour l'analyse en moyenne sur les tries. C'est un moyen les plus directs et élégant pour parvenir à extraire le comportement asymptotique d'une somme harmonique tout en mettant en évidence d'éventuels phénomènes oscillatoires de très faible amplitude.

3.8.1 Définitions

Dans la suite, la notation $\langle \alpha, \beta \rangle$ désignera la bande ouverte du plan complexe $\{ s \in \mathbb{C} \mid \mathrm{Re}(s) \in]\alpha, \beta[\}$.

DÉFINITION 3.11 (TRANSFORMÉE DE MELLIN). *Soit $f :]0, \infty[\to \mathbb{R}$ une fonction localement sommable sur $]0, \infty[$, la transformée de Mellin est définie par*

$$\mathsf{mellin}[f(x); s] = f^\star(s) = \int_0^{+\infty} f(x) x^{s-1} dx$$

La plus grande bande $\langle \alpha, \beta \rangle$ sur laquelle l'intégrale converge est appelée *bande fondamentale* de $f^\star$.

Exemple. La fonction exponentielle a pour transformée la fonction Gamma $\Gamma(s)$ avec pour bande fondamentale $\langle 0, +\infty \rangle$. On définit la fonction exponentielle tronquée $e_b(z)$ d'ordre b pour $b > 0$ par

$$e_b(z) = e^z - 1 + z - \frac{z^2}{2!} + \cdots + \frac{z^{b-1}}{b!}.$$

Cette fonction interviendra dans l'analyse pour $b = 1$ et $b = 2$. On calcule immédiatement (par un changement de variables) que la transformée de $e_b(x)$ est

$$e_b(x) \to \Gamma(s) \qquad s \in \langle -b, -b+1 \rangle .$$

Remarque – En coupant l'intégrale en 2, $\int_0^\infty = \int_0^1 + \int_1^\infty$, on voit que les conditions

$$f(x) \underset{x \to 0^+}{=} O(x^u), \quad f(x) \underset{x \to \infty}{=} O(x^v),$$

garantissent, si $u > v$, que $f^\star(s)$ existe dans la bande $\langle -u, -v \rangle$. Ainsi apparaît le rapport entre le développement asymptotique de f en 0 (resp. en $+\infty$) et la frontière gauche (resp. droite) de la bande fondamentale de $f^\star$.

3.8.2 Propriétés fonctionnelles

Soit $f :]0, \infty[\to \mathbb{R}$ une fonction localement sommable sur $]0, \infty[$ dont la transformée de Mellin admet $\langle \alpha, \beta \rangle$ comme bande fondamentale. On peut établir directement les propriétés suivantes :

(i)	$\mathsf{mellin}[f(\mu x); s] = \mu^{-s} f^\star(s)$	$s \in \langle \alpha, \beta \rangle$, $\mu > 0$
(ii)	$\mathsf{mellin}[\lambda f(x); s] = \lambda f^\star(s)$	$s \in \langle \alpha, \beta \rangle$
(iii)	$\mathsf{mellin}[\sum_{k \in K} \lambda_k f(\mu_k x); s] = (\sum_{k \in K} \lambda_k \mu_k^{-s}) f^\star(s)$	$s \in \langle \alpha, \beta \rangle$, $\mu_k > 0$, K fini
(iv)	$\mathsf{mellin}[f(\frac{1}{x}); s] = f^\star(-s)$	$s \in \langle -\beta, -\alpha \rangle$

(i) et (ii) s'obtiennent facilement grâce à la linéarité de l'intégration et un changement de variable. (iii) découle directement de (i) et (ii). On verra plus loin qu'on peut étendre cette formule sous certaines conditions au cas où K est infini. Enfin, un changement de variable permet d'obtenir (iv). Cette formule permet de ne chercher que le comportement asymptotique en 0^+ (resp. en $+\infty$), puisque le développement asymptotique de $g(x)$ en $+\infty$ (resp. 0^+) se ramène à celui de $f(x) = g(\frac{1}{x})$ en 0^+ (resp. $+\infty$).

3.8.3 Propriétés asymptotiques

Il y a une correspondance précise entre le développement asymptotique d'une fonction en 0 (resp. $+\infty$), et les pôles de la transformée de Mellin dans un demi-plan gauche (resp. droit) par rapport à la bande fondamentale. Chaque terme du développement asymptotique de f de la forme $x^c(\log x)^k$ correspond à un pôle d'ordre $k+1$ de sa transformée $f^\star$ en $s = -c$. Mais cette correspondance sera utilisée dans le sens inverse. Pour déterminer le développement asymptotique de $F(x)$, on calcule la transformée de Mellin $F^\star(s)$ et chaque pôle de $F^\star$ donnera un terme du développement asymptotique de F. Soit une fonction Φ méromorphe en $s = s_0$. La fonction Φ se développe en une série de Laurent au voisinage de s_0

$$\Phi(s) = \sum_{k=-\infty}^{+\infty} c_k (s - s_0)^k.$$

La fonction $\Phi(s)$ a un *pôle* d'ordre r si $r > 0$ et $c_{-r} \neq 0$, et est *holomorphe* en s_0 si $c_k = 0$ pour tout $k < 0$. On définit la *partie singulière* de Φ en $s = s_0$ par

$$\sum_{k=-\infty}^{-1} c_k (s - s_0)^k.$$

DÉFINITION 3.12. *Soit Φ méromorphe sur Ω et $\mathcal{S} \subseteq \Omega$ l'ensemble des pôles de Φ dans Ω. La* partie singulière *de Φ sur Ω est la somme formelle des parties singulières de Φ en chacun des points de $\mathcal{S}$.*

Notation – Si E est la partie singulière de Φ sur Ω, on utilise la notation

$$\Phi(s) \asymp E \qquad (s \in \Omega).$$

Exemple :

$$\frac{1}{s^2(s+1)} \asymp \frac{1}{s+1} + \frac{1}{s^2} - \frac{1}{s} \qquad (s \in \langle -2, 2 \rangle)$$

La notion de partie singulière d'une fonction méromorphe sur un ensemble $\Omega \subseteq \mathbb{C}$ étant définie, on peut énoncer le théorème suivant qui relie l'expression singulière de la transformée de Mellin d'une fonction avec le développement asymptotique en $+\infty$:

THÉORÈME 3.13. *Soit f une fonction continue sur $]0, +\infty[$ dont la transformée de Mellin $f^\star$ admet une bande fondamentale non vide $\langle \alpha, \beta \rangle$. On suppose que*

(i) $f^\star(s)$ admet un prolongement méromorphe sur $\langle \alpha, \gamma \rangle$ avec $\gamma > \beta$, et est analytique sur $\mathrm{Re}(s) = \gamma$.

(ii) Il existe un réel $\eta \in]\alpha, \beta[$, un entier $r > 1$ et une suite réelle $(T_j)_{j\in\mathbb{N}}$ strictement croissante divergeant vers $+\infty$ tels que

$$f^\star(s) = O(|s|^{-r}),$$

sur la réunion des segments $\{s \in \mathbb{C} \mid \mathrm{Re}(s) \in [\eta, \gamma]\,, \mathrm{Im}(s) = T_j\}$, quand $j \to +\infty$.

Si $f^\star(s)$ admet comme partie singulière

$$f^\star(s) \asymp \sum_{(\zeta,k)\in A} d_{\zeta,k} \frac{1}{(s-\zeta)^k} \qquad (s \in \langle \eta, \gamma \rangle),$$

alors le développement asymptotique de $f(x)$ en $+\infty$ est

$$f(x) = \sum_{(\zeta,k)\in A} d_{\zeta,k} \frac{(-1)^k}{(k-1)!} x^{-\zeta} (\log x)^k + O(x^{-\gamma}).$$

Fluctuations périodiques. Un pôle de $f^\star$ en un point $\zeta = \sigma + it$ non réel introduit dans le développement asymptotique un terme de la forme

$$x^{-\zeta} = x^{-\sigma} e^{-it \log x}$$

qui contient une composante périodique en $\log x$ de période $2\pi/t$. Ce sont ces phénomènes de fluctuations qui rendent la transformée de Mellin si utile puisqu'ils sont difficilement accessibles par d'autres méthodes.

3.8.4 Sommes harmoniques

DÉFINITION 3.14. *Une somme de la forme*

$$G(x) = \sum_{k\in K} \lambda_k g(\mu_k x) \qquad \textit{(K fini ou infini)}$$

est appelée somme harmonique. Les ensembles $\{\lambda_k\}$ et $\{\mu_k\}$ forment respectivement l'ensemble des amplitudes et des fréquences. La fonction $g(x)$ est appelée fonction de base de la somme harmonique.

Dans la suite, seuls les développements asymptotiques en $+\infty$ nous intéressent. On va donc seulement s'intéresser aux sommes harmoniques dont les fréquences harmoniques μ_k tendent vers 0. La série de Dirichlet associée à la somme harmonique est

$$\Lambda(s) = \sum_{k\in K} \lambda_k {\mu_k}^{-s}.$$

La transformée de Mellin d'une somme harmonique finie s'écrit

$$G^\star(s) = \Lambda(s) g^\star(s).$$

On peut étendre cette formule du produit aux sommes harmoniques infinies sous les conditions du théorème des sommes harmoniques. Ce théorème permet de garantir la convergence d'une somme harmonique et d'en obtenir son développement asymptotique.

THÉORÈME 3.15 (SOMMES HARMONIQUES). *Soit $G(x)$ une somme harmonique,*

$$G(x) = \sum_{k\in K} \lambda_k g(\mu_k x),$$

telle que :

- *g est continue sur $]0,+\infty[$ et $g^\star$ a pour bande fondamentale $\langle \alpha, \beta\rangle$;*
- *$\Lambda(s) = \sum_{k\in K} \lambda_k {\mu_k}^{-s}$ la série de Dirichlet associée à $G(x)$ a un demi-plan de convergence simple $\mathrm{Re}(s) < \sigma$.*

Supposons de plus que

(i) $\sigma > \alpha$, c'est à dire que le demi-plan de convergence simple de $\Lambda(s)$ a une intersection non vide avec la bande fondamentale $\langle \alpha, \beta\rangle$. Notons $\beta' =\in (\beta, \sigma)$.

(ii) Il existe un nombre réel $\gamma > \beta$ pour lequel $g^\star$ et Λ admettent un prolongement méromophe sur $\langle \alpha, \gamma\rangle$ et soient analytiques sur $\mathrm{Re}(s) = \gamma$.

(iii) Il existe un réel $\eta \in]\alpha, \beta'[$ et une suite réelle $(T_j)_{j\in\mathbb{N}}$ strictement croissante divergeant vers $+\infty$ tels que

$$g^\star(s) = O(|s|^{-k}),\ \textit{pour tout entier } k > 0,$$
$$\Lambda(s) = O(|s|^{r}),\ \textit{pour un entier } r > 0,$$

sur la réunion des segments $\{s \in \mathbb{C} \mid \mathrm{Re}(s) \in [\gamma, \eta]\ , \mathrm{Im}(s) = T_j\}$ quand $j \to +\infty$.

Alors la somme harmonique $G(x)$ converge sur $]0,+\infty[$ et sa transformée $G^\star(s)$ est bien définie sur $\langle \alpha, \beta'\rangle$ et s'écrit $G^\star(s) = \Lambda(s)g^\star(s)$. Si $G^\star(s)$ admet de plus comme partie singulière

$$G^\star(s) \asymp \sum_{(\zeta,k)\in A} d_{\zeta,k} \frac{1}{(s-\zeta)^k} \qquad (s \in \langle \gamma, \eta\rangle),$$

alors le développement asymptotique de $G(x)$ en $+\infty$ est

$$G(x) = \sum_{(\zeta,k)\in A} d_{\zeta,k} \frac{(-1)^k}{(k-1)!} x^{-\zeta} (\log x)^k + O(x^{-\gamma}).$$

3.8.5 Application

Comme exemple d'application de la transformée de Mellin, examinons les expressions (3.9), (3.10) et (3.11) obtenues respectivement pour la taille s_n, la longueur de cheminement ℓ_n et la hauteur h_n d'un trie binaire construit avec une source symétrique sans mémoire symétrique

$$s_n = \sum_{k\geq 0} 2^k (1 - (1 - \frac{1}{2^k})^n - \frac{n}{2^k}(1 - \frac{1}{2^k})^{n-1}) \tag{3.17}$$

$$\ell_n = n \sum_{k\geq 0} (1 - (1 - \frac{1}{2^k})^{n-1}) \tag{3.18}$$

$$h_n = \sum_{k\geq 0} \left[1 - n!\,[z^n](1 + z/2^k)^{2^k}\right]. \tag{3.19}$$

Afin de ne pas alourdir inutilement cette présentation, on va ici faire appel à une simplification qui sera pleinement justifiée au chapitre 8. Cette simplification consiste à opérer la correspondance

$$(1-a)^n \mapsto e^{-an} \tag{3.20}$$

$$n(1-a)^{n-1} \mapsto ne^{-na} \tag{3.21}$$

$$n!\,[z^n](1+z/2^k)^{2^k} \mapsto e^{-n}\left(1+\frac{n}{2^k}\right)^{2^k}. \tag{3.22}$$

Derrière ce «stratagème» se cache en fait le passage à un modèle de Poisson où les calculs peuvent être menés à terme plus simplement (voir chapitre 8). Les relations (3.20) et (3.21) sont des équivalents. La méthode du point de col permet d'obtenir la relation (3.22) (cela sera détaillé lors de la présentation de cette méthode au chapitre 8).

Longueur de cheminement externe

On va étudier la somme harmonique correspondant à ℓ_n

$$F(x) = \sum_{k\geq 0} 2^k \frac{x}{2^k}(1-e^{x/2^k}), \tag{3.23}$$

de fonction de base $f(x) = x(1-e^{-x})$, de fréquences $\{\mu_k = 2^{-k}\}_{k\geq 0}$ et d'amplitudes $\{\lambda_k = 2^k\}_{k\geq 0}$. On calcule la transformée de Mellin

$$f^\star(s) = -\Gamma(s+1) \qquad (s \in \langle -2, -1\rangle).$$

La série de Dirichlet associée à $F(x)$ est

$$\Lambda(s) = \sum_{k\geq 0} 2^k 2^{ks} = \sum_{k=0}^{\infty} (2^{s+1})^k = \frac{1}{1-2^{s+1}}$$

La série de Dirichlet $\Lambda(s)$ a un demi-plan de convergence simple $\mathrm{Re}(s) < -1$. Les pôles de Λ sont situés sur une droite verticale $\mathrm{Re}(s) = -1$ aux points

$$\chi_k = -1 + \frac{2\mathrm{i}k\pi}{\log 2},$$

pour $k \in \mathbb{Z}$. La figure 3.1 précise les conditions d'application du théorème (les pôles sont représentés par des points). Appliquons le théorème des sommes harmoniques

- $f^\star$ a pour bande fondamentale $\langle -2, -1\rangle$;
- Λ a pour demi-plan de convergence simple $\mathrm{Re}(s) < -1$ d'intersection non vide avec $\langle -2, -1\rangle$;
- Λ et $f^\star$ admettent un prolongement méromorphe sur $\mathbb{C}$ et sont analytiques sur $\mathrm{Re}(s) = \delta$ pour tout réel $\delta < -1$;
- Soit $\delta < 1$. On considère la suite (T_j) définie par

$$T_j = \frac{(2k+1)\pi}{\log 2}.$$

Alors sur la réunion des segments, défini pour éviter les pôles de Λ, on a

$$\{s \in \mathbb{C} \mid \mathrm{Re}(s) \in [-\frac{3}{2}, \delta] \; et \; \mathrm{Im}(s) = T_j\},$$

qui est défini pour éviter les pôles de Λ, on a

$$|\Lambda(\sigma + \mathrm{i}T_j)| = |\frac{1}{1+2^{1+\sigma\mathrm{i}T_j}}| \leq \frac{1}{1+2^{1-3/2}} = O(|s|^0).$$

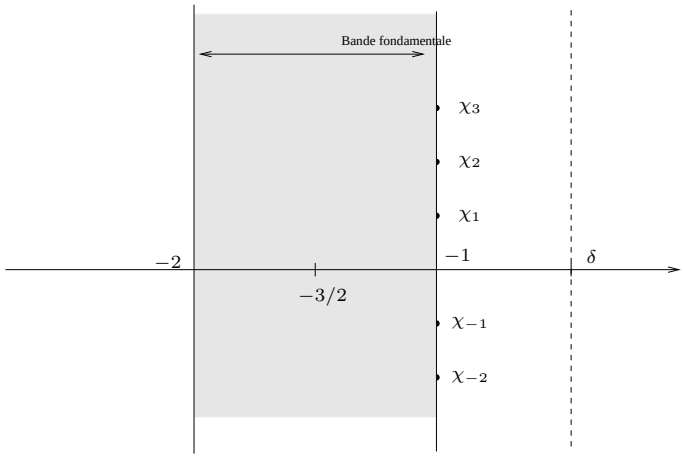


FIG. 3.1 – Situation des pôles et de la bande fondamentale dans le plan complexe.

De plus, la version complexe de la formule de Stirling

$$|\Gamma(\sigma+\mathrm{i}t)| \sim \sqrt{2\pi}|t|^{\sigma-1/2}e^{-\pi|t|/2} \qquad (t\to+\infty),$$

permet d'obtenir pour tout $r>1$, quand $j\to+\infty$, que

$$|f^\star(\sigma+\mathrm{i}T_j)| = |\Gamma(1+\sigma+\mathrm{i}T_j)| = O(|s|^{-r}).$$

Le théorème s'applique : $F(x)$ converge sur $]0,+\infty[$. La partie singulière de $F^\star(s)=\Lambda(s)F^\star(s)$ est

$$F^\star(s) \asymp \frac{1}{\log 2}\frac{1}{(s-1)^2} - (\frac{1}{2}+\frac{\gamma}{\log 2})\frac{1}{s-1} + \sum_{k\in\mathbb{Z}^\star}\frac{1}{\log 2}\frac{\Gamma(1-\chi_k)}{s-\chi_k} \qquad (s\in\langle -3/2,\delta\rangle),$$

où γ désigne la constante d'Euler. Il faut pour la calculer se rappeler que

$$\Gamma(s) = \frac{1}{s} - \gamma + O(s) \qquad (s\to 0)$$
$$(1-2^{-s})^{-1} = \frac{1}{\log 2}\frac{1}{s} + \frac{1}{2} + O(s) \qquad (s\to 0)$$

On obtient alors le développement de $F(x)$ quand $x\to+\infty$ et donc le développement asymptotique de ℓ_n

$$\ell_n = \frac{1}{\log 2}n\log n + (\frac{1}{2}+\frac{\gamma}{\log 2})n + \frac{n}{\log 2}P(\log_2 n) + O(n^{-\delta}).$$

Le polynôme $P(\log_2 x)$ regroupe les termes imaginaires liés aux χ_k.

$$P(\log_2 x) = \sum_{k\in\mathbb{Z}^\star}\Gamma(\frac{2\mathrm{i}k\pi}{\log 2})e^{-2\mathrm{i}k\pi\log_2(x)}.$$

Du fait de la forte décroissance de $\Gamma(\frac{2\mathrm{i}k\pi}{\log 2})$ quand $k\to+\infty$, $|P(\log_2 x)|$ est majoré par 10^{-6}.

Nombre de nœuds internes

Le développement asymptotique du nombre de nœuds internes se calcule de la même façon que pour la longueur de cheminement externe. On considère la somme harmonique

$$F(x) = \sum_{k\geq 0} 2^k(1 - e^{-x/2^k}(1 + x/2^k)),$$

de fonction de base $f(x) = (1 - e^{-x}(1 + x))$, de fréquences $\mu_k = 2^{-k}$ et d'amplitudes $\lambda_k = 2^k$. L'espérance s_n est égale à $F(n)$. On calcule la transformée de Mellin de f

$$f^\star(-s) = -(s + 1)\Gamma(s) \qquad (s \in \langle -2, 0\rangle).$$

La série de Dirichlet associée à $F(x)$ est

$$\Lambda(s) = \frac{1}{1 - 2^{1+s}} \qquad (s \in \langle -\infty, -1\rangle).$$

Le théorème des sommes harmoniques s'applique encore. Soit $\delta > 0$. Cette fois ci, $F^\star$ n'a que des pôles simples sur $\left\langle -\frac{3}{2}, \delta\right\rangle$ situés en 0 et aux points

$$\chi_k = -1 + \frac{2\mathrm{i}k\pi}{\log 2}, \text{ pour } k \in \mathbb{Z}.$$

La partie singulière de $F^\star(s) = \Lambda(s)f^\star(s)$ s'écrit

$$F^\star(s) \asymp \frac{1}{\log 2}\frac{1}{s-1} + \frac{1}{s} + \sum_{k\in\mathbb{Z}^\star} \frac{1}{\log 2}\frac{(\chi_k - 1)\Gamma(-\chi_k)}{s - \chi_k} \qquad (s \in \langle -3/2, \delta\rangle).$$

On a besoin des 2 développements suivants

$$\Gamma(s) = -\frac{1}{s+1} + O(1) \qquad (s \to -1)$$
$$(1 - 2^{-s})^{-1} = \frac{1}{\log 2}\frac{1}{s} + \frac{1}{2} + O(s) \qquad (s \to 0).$$

On obtient immédiatement le développement de $F(x)$ quand $x \to \infty$ pour tout $c > 1$

$$F(x) = \frac{1}{\log 2}x + 1 - \frac{x}{\log 2}Q(\log_2 x) + O(x^{-c}).$$

Le polynôme $Q(\log_2 x)$ est un polynôme regroupant les termes imaginaires liés aux χ_k. $Q(u)$ est périodique de période 1, et d'amplitude négligeable devant 1 (mais intervient contrairement au cas de la longueur de cheminement externe dans le terme principal du développement asymptotique).

Hauteur

La hauteur ne se traite pas aussi simplement puisqu'elle ne s'exprime pas directement comme une somme harmonique.

$$h_n = \sum_{k\geq 0} \left[1 - \left(e^{-n/2^k}(1 + n/2^k)\right)^{2^k}\right].$$

On se limite à une majoration de la quantité précédente[3] grâce à l'inégalité

$$(1+x)e^{-x} \geq e^{-\frac{x^2}{2}}.$$

On en déduit

$$h_n \leq \sum_{\ell \geq 0} (1 - e^{-n^2/2^{k+1}}).$$

On examine la somme harmonique

$$F(x) = \sum_{k \geq 0} f(\mu_k x) \text{ avec } f(x) = 1 - e^{-x^2} \text{ et } \mu_k = 2^{(k+1)/2}.$$

On calcule la transformée de Mellin de $f(x)$ (de bande fondamentale $\langle -2, 0\rangle$) et la série de Dirichlet (de demi-plan de convergence $\langle -\infty, 0\rangle$)

$$f^\star(s) = -\frac{1}{2}\Gamma(s/2) \quad \text{et} \quad \Lambda(s) = \frac{2^{s/2}}{1-2^{s/2}}.$$

Les deux fonctions admettent bien un prolongement méromorphe sur $\mathbb{C}$. Soit $\delta > 0$. La partie singulière de $F^\star(s)$ sur $\langle -1/2, \delta\rangle$ est

$$F^\star(s) \asymp \frac{2}{\log 2}\frac{1}{s^2} + (\frac{1}{2} - \frac{\gamma}{\log 2})\frac{1}{s} + \sum_{k \in \mathbb{Z}^\star} \frac{1}{\log 2}\frac{\Gamma(-\chi_k)}{s-\chi_k} \qquad (s \in \langle -1/2, \delta\rangle),$$

où γ désigne la constante d'Euler et $\chi_k = 2\mathrm{i}k\pi/\log 2$. On obtient le développement asymptotique

$$h_n \leq \frac{2}{\log 2}\log n + (\frac{-1}{2} + \frac{\gamma}{\log 2}) - \frac{1}{\log 2} R(\log n^2/2) + O(n^{-\delta}).$$

Encore une fois $R(\log_2 x)$ est périodique d'amplitude négligeable devant 1.

Remarque. On peut montrer en utilisant une méthode plus fine que la hauteur a effectivement pour espérance en $\frac{2}{\log 2}\log(x) + O(1)$. La majoration utilisée est une «bonne» majoration. Nous le montrerons au chapitre 8 avec un modèle de source beaucoup plus général.

[3] La hauteur sera traitée en détail avec un modèle de source plus général au chapitre 8.

Chapitre 4

Sources dynamiques probabilisées

Sommaire

Nous abordons dans ce chapitre le cadre général des *sources dynamiques probabilisées*. Tout d'abord, nous introduisons deux types de sources dynamiques *symboliques*, nommés basiques et Markoviens. De tels mécanismes sont reliés aux *systèmes dynamiques* définis à l'aide d'applications expansives sur l'intervalle $[0, 1]$ (voir à ce propos [5, 68, 87]). Le modèle peut encore être étendu en considérant une fonction de densité (analytique) de l'intervalle $[0, 1]$, ce qui définit un nouveau type de source : *les sources dynamiques probabilisées*. Enfin, nous présentons les notions d'intervalle fondamental et de mesure fondamentale, puis introduisons deux caractéristiques de la source, l'entropie et la probabilité de coïncidence.

Les sources dynamiques sont introduites et étudiées dans [102] et des notions similaires jouent un rôle important pour l'analyse d'algorithmes autour de l'algorithme d'Euclide en théorie des nombres [100, 99, 101].

4.1 Comment produire des symboles ? Les sources classiques.

On considère ici une source qui produit un flot infini de symboles. On cherche un modèle qui s'approche le plus possible de la «réalité» tout en restant utilisable théoriquement (le modèle n'est pas «vide» et permet de mener des calculs à terme).

La production de symboles est un phénomène *discret*. On peut ainsi s'imaginer qu'à chaque coup d'horloge, un nouveau symbole est émis par la source. Le choix du symbole à émettre peut prendre en compte un grand nombre de paramètres. Nous décrivons ici deux sources classiques dont le mécanisme est probabiliste : les sources sans mémoire et les chaînes de Markov.

4.1.1 Sources sans mémoire

Le principe d'une source sans mémoire est simple. On se donne un alphabet $\mathcal{M}$ et les fréquences d'apparition des symboles $\{p_i\}_{i\in\mathcal{M}}$. Puis, on émet à chaque coup d'horloge le symbole i avec la probabilité p_i. Par définition, la source sans mémoire ne tient aucun compte des symboles précédemment émis : il n'y a aucune dépendance entre deux symboles émis par la source.

Tous les exemples émaillant cette partie sont tirés du livre de Dominic Welsh [104]. Pour la langue anglaise avec un alphabet de taille 27 (les lettres plus le caractère espace), une première tentative (grossière) de modélisation, dite approximation d'ordre 0, consiste à émettre chaque lettre avec une probabilité $\frac{1}{27}$. Un exemple typique d'un mot produit par une telle source est (d'après [104])

DM QASCJDGFOZYNX ZSDZLXIKUD.

La deuxième étape, naturelle, consiste à considérer des probabilités non uniformes calculées à partir d'un corpus. On obtient alors un mot qui «ressemble» déjà plus d'un point de vue syntaxique à une phrase naturelle en anglais

OR L RW NILI E NNSBATEI.

4.1.2 Chaînes de Markov

Le prochain échelon à gravir consiste à prendre en compte les dépendances entre les lettres. En effet, la lettre 'T' en anglais a toutes les chances d'être suivie d'un 'H'. En français, la lettre 'Q' sera souvent suivie de la lettre 'U'. Les chaînes de Markov du premier ordre (car il est évidemment possible d'examiner les dépendances plus lointaines, non réduites à deux lettres consécutives) permettent de prendre en compte ce type de modèle.

Considérons les probabilités conditionnelles calculées à partir des fréquences de paires de lettres successives (ou digrammes)

$$p_{i|j} = p_{ji}/p_j$$

où p_{ji} est la probabilité du digramme ji (la lettre j suivie de la lettre i) et $p_{i|j}$ est la probabilité conditionnelle d'avoir un symbole i lorsque le symbole précédent est j. Une chaîne de Markov est donc définie à partir d'une matrice de transition $\Pi = (p_{i|j})$ et un vecteur π_0 de probabilités initiales. Un vecteur $\pi = (p_i)$ de probabilités est tel que $p_i \geq 0$ pour tout i et $\sum p_i = 1$. Ces matrices ont beaucoup de propriétés et sont abondamment étudiées (voir [29] par exemple pour une présentation plus détaillée). Par exemple, pour $p = (p_i)$ un vecteur de probabilités, le vecteur $\Pi \times p$ est également un vecteur de probabilités. Les coefficients des itérés de la matrice Π ont une signification importante. En effet le coefficient (i, j) de Π^k (ligne i et colonne j) correspond à la probabilité conditionnelle d'avoir un mot de longueur k qui commence par la lettre j et finisse par la lettre i.

Pour générer un texte qui obéisse à ce modèle (issu d'un corpus), on peut utiliser une méthode de type Monte Carlo due à Shannon. Cette méthode permet d'éviter le calcul des probabilités à partir du corpus. On choisit un texte au hasard et on pointe une lettre aléatoirement. Supposons que cette lettre soit 'B'. On pointe à nouveau au hasard un endroit du texte et l'on parcourt le texte jusqu'à rencontrer la lettre 'B'. Le symbole à émettre est alors le symbole le suivant immédiatement. Ce symbole devient le symbole courant, et on itère le processus.

Grâce à cette méthode, on génère par exemple le texte suivant qui correspond à une chaîne de Markov du premier ordre

OUCTIE IN ARE AMYST TE TUSE SOBE CTUSE.

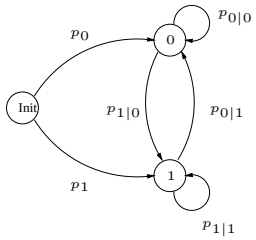


FIG. 4.1 – Automate (non déterministe) correspondant à une chaîne de Markov sur un alphabet $\{0, 1\}$ avec des probabilités initiales p_0, p_1 et une matrice de transition $(p_{i|j})$.

Plutôt que de prendre des dépendances sur deux lettres, on peut considérer une «fenêtre» sur les derniers caractères afin d'émettre le prochain symbole (la méthode de Shanon s'adapte facilement). On obtient alors des approximations d'ordre supérieur correspondant à des chaînes de Markov d'ordre supérieur. Une approximation par une chaîne de Markov d'ordre 2 donne la phrase

HE AREAT BEIS HEDE THAT WISHBOUT SEED DAY OFTE,
AND HE IS FOR THAT MINUMB LOOTS WILL AND GIRLS,
A DOLL WILL IS FRIECE ABOARDARICE STRED SAYS,

(Source : Runyon on Broadway)

En faisant preuve de beaucoup d'imagination et en acceptant les néologismes au sens trouble, on peut commencer à trouver du sens à cette phrase !

C'est peut-être plus frappant encore avec des langues qui nous sont familières sans que pour autant nous les lisions couramment. Ainsi prenant comme corpus le texte «*de Senecute*» de Ciceron, nous obtenons pour le latin les approximations

IENEC FES VIMONILLITUM M ST ER PEM ENIM PTAUL

(Chaîne de Markov premier ordre),

SENECTOR VCI QUAEMODOMIS SE NON
FRATURDIGNAVIT SINE VELIUS

(Chaîne de Markov deuxième ordre).

Une définition équivalente des chaînes de Markov est de les visualiser grâce à des automates non déterministes. L'ensemble des états est constitué de l'ensemble des symboles plus un état initial. De chaque état j partent les arcs vers chacun des états i, l'arc correspondant étant emprunté avec la probabilité $p_{i|j}$ (voir figure 4.1). De l'état initial partent des arcs correspondant aux probabilités initiales $\{p_i\}$. On part de l'état initial. Puis, à chaque fois que l'on arrive dans un état on émet le symbole correspondant.

La tâche qui consiste à modéliser une source en partant d'un texte réel (que ce soit un corpus en langue naturelle, ou encore des numéros de cartes de crédit) est ardue. Le modèle idéal tiendrait compte de l'ensemble des caractères déjà émis pour choisir le prochain symbole, c.-à-d. qu'il considérerait pour un alphabet $\mathcal{M}$ les ensembles $\{p_{\boldsymbol{w}}\}_{\boldsymbol{w}\in\mathcal{M}^k}$ des probabilités d'apparition d'un préfixe $\boldsymbol{w}$ de longueur k

$$p_{\boldsymbol{w}} = \Pr\{\boldsymbol{v} \in \mathcal{M}^{\mathbb{N}} \text{ a pour préfixe } \boldsymbol{w}\}.$$

Bien sûr, on a l'égalité pour tout $k \geq 0$

$$\sum_{\boldsymbol{w} \in \mathcal{M}^k} p_{\boldsymbol{w}} = 1.$$

Les sources dynamiques fournissent un cadre général qui permet de modéliser à la fois les sources sans mémoire, les chaînes de Markov mais également d'autres types de sources comme par exemple la source en fraction continue qui prend en compte l'ensemble du passé pour décider du prochain symbole à émettre.

4.2 Mécanisme basé sur les sources dynamiques

4.2.1 Sources dynamiques symboliques de base

Dans le contexte de la théorie de l'information, une source est un mécanisme qui produit des mots infinis sur un alphabet $\mathcal{M}$. Dans un premier temps, ce sont les sources associées aux systèmes dynamiques de base (pour lequel le mécanisme est le même à chaque étape) qui nous intéressent.

DÉFINITION 4.1 (SOURCE SYMBOLIQUE DE BASE). *Une source dynamique de base est définie à l'aide de quatre éléments :*

(a) *Un* alphabet $\mathcal{M}$ *inclus dans* $\mathbb{N}$ *fini ou dénombrable.*

(b) *Une* partition topologique *de* $\mathcal{I} :=]0,1[$ *avec des intervalles ouverts disjoints* $\mathcal{I}_m$*, i.e.,* $\overline{\mathcal{I}} = \bigcup_{m \in \mathcal{M}} \overline{\mathcal{I}}_m$.

(c) *Une application de* codage σ *constante et égale à* m *sur chaque intervalle* $\mathcal{I}_m$.

(d) *Une application de* décalage T *dont la restriction* $T_{[m]}$ *à chaque intervalle* $\mathcal{I}_m$ *est une bijection analytique de* $\mathcal{I}_m$ *sur* $\mathcal{I}$*. On note* h_m *l'application inverse de* $T_{[m]}$ *restreinte à l'intervalle* $\mathcal{I}_m$ *et* $\mathcal{H} := \{h_m,\ m \in \mathcal{M}\}$ *l'ensemble des branches inverses de* T*. On suppose qu'il existe un voisinage complexe* $\mathcal{V}$ *de* $\overline{\mathcal{I}}$ *sur lequel l'ensemble* $\mathcal{H}$ *satisfait les conditions suivantes :*

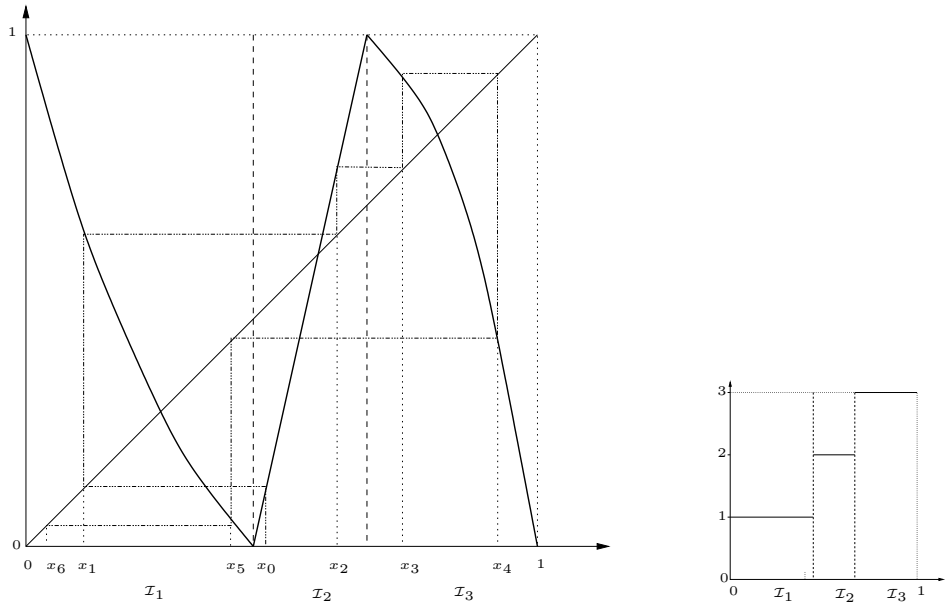
- (d_1) *Les applications* h_m *admettent une extension holomorphe sur* $\mathcal{V}$*, et envoient strictement* $\mathcal{V}$ *dans* $\mathcal{V}$ *(i.e.,* $h(\mathcal{V}) \subsetneq \mathcal{V}$*) ;*
- (d_2) *Les applications* $|h'_m|$ *admettent des extensions holomorphes, notées* $\widetilde{h}_m$*, sur* $\mathcal{V}$ *et il existe des nombres réels* $\delta_m < 1$ *pour lesquels* $0 < |\widetilde{h}_m(z)| \leq \delta_m$ *pour* $z \in \mathcal{V}$ *;*
- (d_3) *Il existe un nombre réel* $\gamma < 1$ *pour lequel la série* $\sum_{m \in \mathcal{M}} \delta_m^s$ *converge pour* $\mathrm{Re}(s) > \gamma$.

Remarques :

1. une telle source dynamique est dite *de base* en vertu des relations $T(\mathcal{I}_m) = \mathcal{I}$. Dans la littérature, il est en général seulement demandé que l'image $T(\overline{\mathcal{I}}_m)$ soit une union d'éléments $\overline{\mathcal{I}}_j$ de la partition.
2. Les conditions (d_1) et (d_2) expriment le fait que les branches inverses h_m sont des contractions, ou de façon équivalente que l'application T est expansive.
3. La condition (d_3) est trivialement vraie dès lors que l'alphabet est fini (avec $\gamma = -\infty$). Cette condition n'intervient que dans les cas des alphabets infinis.
4. Dans les faits, pour que nos méthodes s'appliquent, il est suffisant que l'application obtenue en itérant un nombre fixé de fois T vérifie les conditions (d). Par exemple, l'application de décalage T associée aux fractions continues ne satisfait pas les conditions (d_1) et (d_2) mais son itérée seconde T^2, elle, les satisfait.
5. La fonction f est expansive (respectivement contractante) sur $\mathcal{D}$ ssi $\mathcal{D} \subset f(\mathcal{D})$ (resp. $f(\mathcal{D}) \subset \mathcal{D}$). La condition (d_1) implique que h_m est strictement contractante sur $\mathcal{V}$. La

condition (d_2) exprime la condition pour qu'une fonction dérivable soit strictement contractante sur $[0, 1]$.

6. L'application de décalage T est définie presque partout. Il est important pour la suite que l'ensemble sur lequel T n'est pas défini soit de mesure nulle.



(i) Application de décalage T

(ii) Application de codageσ

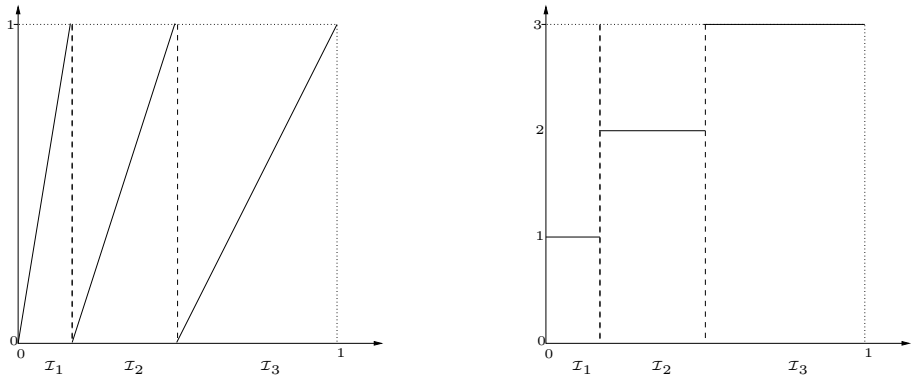
FIG. 4.2 – Exemple de mécanisme de source avec (i) l'application T de décalage et (ii) l'application de codage σ. L'alphabet est $\mathcal{M} = \{1, 2, 3\}$, la partition $\{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\}$. La suite d'itérés associée à $x = x_0$ est $(x_0, x_1, x_2, x_3, x_4, x_5, x_6, \dots)$, ce qui produit le mot $M(x) = (2, 1, 2, 3, 3, 1, 1, \dots)$.

Le lecteur non familier avec la théorie des applications expansives et des systèmes dynamiques symboliques peut examiner la définition que nous avons présentée à la lumière de l'exemple du système de représentation en base 2 usuel. Dans ce cas, l'alphabet est $\mathcal{M} = \{0, 1\}$, le décalage $T(x)$ est $\{2x\}$ (où $\{\cdot\}$ désigne la partie rationnelle), la partition de l'intervalle $\mathcal{I} = [0, 1[$ est $\{\mathcal{I}_0, \mathcal{I}_1\} = \{[0, \frac{1}{2}[, [\frac{1}{2}, 1[\}$, et le codage se fait au moyen de l'application σ constante sur chaque intervalle de la partition $\sigma(\mathcal{I}_0) = 0$, $\sigma(\mathcal{I}_1) = 1$.

Les mots émis par la source sont alors produits comme suit (voir figure 4.2). L'application $T : \mathcal{I} \to \mathcal{I}$ (définie presque partout) est utilisée pour itérer le processus, comme un opérateur de décalage. L'application $\sigma : \mathcal{I} \to \mathcal{M}$ est utilisée pour le codage. Elle détermine le symbole à émettre. La source peut également être vue en termes d'états. Les états sont les lettres de l'alphabet. Chaque itération de l'application de décalage T fait passer dans un nouvel état (codé par σ). Le mot $M(x)$ de $\mathcal{M}^{\mathbb{N}}$ associé au réel $x \in \mathcal{I}$ est alors formé de la suite de symboles

$$M(x) := (M_1(x), M_2(x), \dots, M_k(x), \dots), \tag{4.1}$$

où le k^{e} symbole $M_k(x)$ de $M(x)$ est $\sigma(T^{k-1}x)$. Ainsi le préfixe de $M(x)$ garde l'historique des états par lesquels on est passé.



(i) Application de décalage T (ii) Application de codage σ

FIG. 4.3 – Exemple de mécanisme de source sans mémoire associée aux probabilités $(\frac{1}{6}, \frac{1}{3}, \frac{1}{2})$.

Le nombre de branches de T est égal au cardinal de l'alphabet, et les symboles de l'alphabet sont utilisés pour coder les branches de T, notées $T_{[m]}$ (ou de façon équivalente les branches inverses h_m de T).

Nous pouvons faire le lien entre les applications T et σ et les applications $\underline{T}$ et $\underline{\sigma}$ définies en 2.1.1. Elles sont en effet très proches. La différence essentielle se situe au niveau du domaine de définition (les mots pour les premiers et les réels de $\mathcal{I}$ pour le second). L'application σ donne le premier symbole du codage et T correspond à une opération de décalage. Avec nos notations, on a

$$\underline{\sigma}(M(x)) = \sigma(x), \qquad \underline{T}(M(x)) = M(T(x)).$$

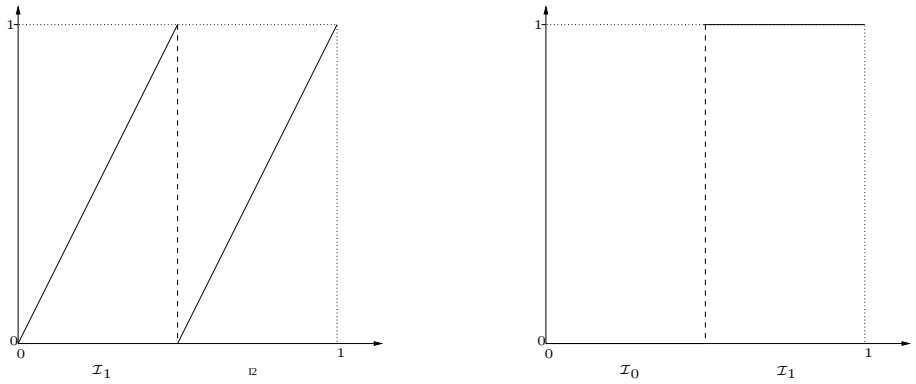
Sources sans mémoire. Toutes les sources sans mémoire entrent dans le cadre des sources dynamiques de base. Une source est dite sans mémoire lorsque les variables aléatoires M_k sont indépendantes et suivent la même loi. La source sans mémoire associée à une séquence de probabilités $P = (p_m)_{m\in\mathcal{M}}$ (finie ou dénombrable) est la source où tous les symboles sont indépendants et suivent une loi multinomiale[1] de paramètre $(p_m)_{m\in\mathcal{M}}$. La partition topologique correspondante de $\mathcal{I}$ est alors définie par

$$\mathcal{I}_m :=]q_m, q_{m+1}[, \qquad \text{où } q_m = \sum_{j<m} p_j,$$

et la restriction $T_{[m]}$ de T à $\mathcal{I}_m$ est affine. On peut donc choisir $T_{[m]}$ croissante ou décroissante en posant $\big(T_{[m]}(q_m), T_{[m]}(q_{m+1})\big) = (0,1)$ ou $(1,0)$. Un exemple de telle source est la source sur un alphabet à trois symboles avec des probabilités d'émission $(\frac{1}{6}, \frac{1}{3}, \frac{1}{2})$ (voir la figure 4.3). Un autre exemple de source est la source associée à l'alphabet $\mathbb{N}^*$ et aux probabilités $\{\frac{1}{2^m}\}_{m\geq 1}$.
Le cas des systèmes de numération en base b est important et est défini par

$$T(x) = \{bx\}, \qquad \sigma(x) = \lfloor bx \rfloor,$$

[1]Les sources sans mémoire produisent des symboles selon des lois de distribution binomiales ou multinomiales et il est de pratique courante de les nommer «sources de Bernoulli». Cette thèse met également en jeu un modèle probabiliste dit de Bernoulli dans lequel le cardinal d'un ensemble est fixé (par opposition au modèle de Poisson). Pour éviter toute confusion, le terme Bernoulli sera utilisé pour désigner ce type de modèle aléatoire, tandis qu'on utilisera source sans mémoire pour les sources.



(i) Application de décalage T (ii) Application de codage σ

FIG. 4.4 – Exemple de mécanisme de source associé au développement binaire.

où $\lfloor u \rfloor$ est la partie entière de u et $\{u\} = u \bmod 1 = u - \lfloor u \rfloor$ est la partie fractionaire de u. Ces transformations permettent de calculer le développement en base b de x sous la forme

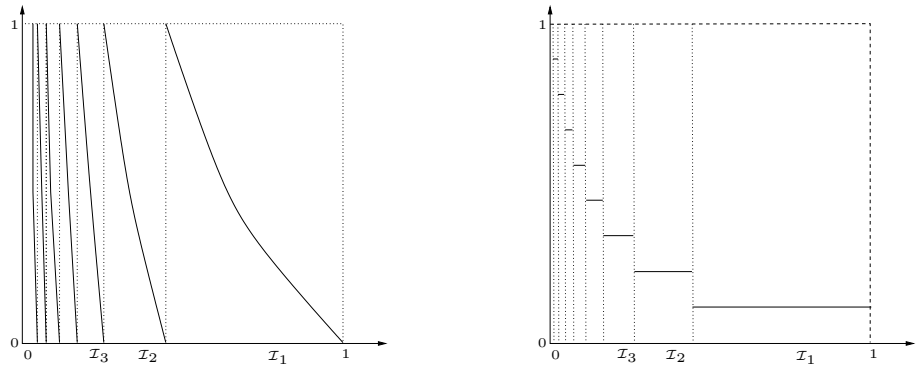
$$x = \sum_{j=1}^{\infty} m_j b^{-j}, \text{ avec } m_j = \sigma(T^j(x)), \tag{4.2}$$

et sont associées aux sources sans mémoire symétriques (i.e., sources sans mémoire où tous les p_j sont égaux).
L'exemple le plus utilisé est le système de numération en base 2 qui consiste à exprimer un nombre en binaire. La figure 4.4 donne les applications de décalage T et de codage σ de la source.

Numération en fraction continue. Le cadre général des sources dynamiques permet de décrire des sources différentes qui, par opposition aux sources précédentes, ont «de la mémoire». Il suffit pour cela de considérer une application T avec au moins une branche non affine. En effet tant que la branche est affine, elle effectue un «zoom» uniforme de $\mathcal{I}_m$ vers $\mathcal{I}$. En un certain sens, c'est la dérivée $T'(x)$ qui permet de garder en mémoire l'historique. Le mécanisme fondé sur la numération en fraction continue considère les applications de décalage et de codage

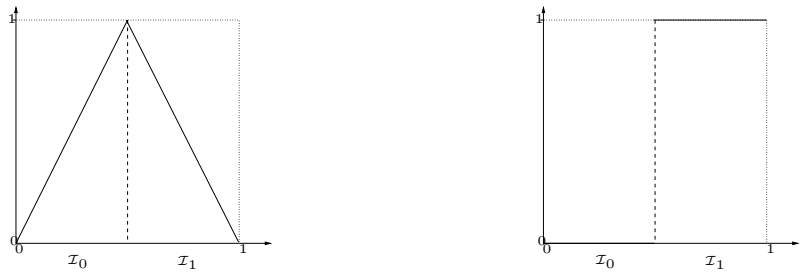
$$T_{CF}(x) = \{\frac{1}{x}\}, \qquad \sigma_{CF}(x) = \lfloor \frac{1}{x} \rfloor.$$

L'alphabet est l'ensemble des entiers naturels $\mathbb{N}$, la partition topologique de $\mathcal{I}$ est définie par $\mathcal{I}_m :=]1/(m+1), 1/m[$, et la restriction de T à $\mathcal{I}_m$ est l'homographie $T(x) := (1/x) - m$ (voir la figure 4.5), En itérant cette transformation, le développement en fraction continue de x est obtenu. Ce système de numération est notamment utilisé en algorithmique géométrique pour déterminer le signe d'un déterminant 2×2 (voir également HAKMEM).
Les branches inverses sont toutes des homographies h_m définies par $h_m(x) := 1/(x+m)$. Bien que la première branche h_1 ne satisfasse pas les conditions (d_1) et (d_2), ces conditions sont satisfaites par l'ensemble des homographies $h_m \circ h_n$. Les branches avec des points fixes perturbent le système dynamique. Avec un peu d'attention, on peut malgré tout souvent éviter les cas «pathologiques». Quand une «mauvaise» homographie survient, on la considère conjointement avec une «bonne» de façon à éviter ces problèmes liés aux points fixes.



(i) Application de décalage T (ii) Application de codage σ

FIG. 4.5 – Exemple de mécanisme de source associée au développement en fraction continue.



(i) Application de décalage T (ii) Application de codage σ

FIG. 4.6 – Exemple de mécanisme de source associé au code de Gray.

Homocline ou hétérocline ? Notre définition d'une source dynamique probabilisée n'exige nullement que toutes les branches soient toutes croissantes ou toutes décroissantes. Nous introduisons le terme *homocline* pour désigner le cas où toutes les branches sont du même type (croissantes ou décroissantes). Le terme *hétérocline* désignera le reste des configurations. Les systèmes de numération binaire et en fraction continue sont homoclines. Le code de Gray (où l'on passe d'un code au suivant on modifiant juste un bit) correspond pour les codes de longueur 3 aux mots

$$000, 001, 011, 010, 110, 111, 101, 100.$$

Cette source hétérocline considère l'opérateur de décalage (voir figure 4.6)

$$T(x) = 2x \text{ si } x \in [0, \frac{1}{2}[, \qquad T(x) = 2 - 2x \text{ si } x \in [\frac{1}{2}, 1[.$$

4.2.2 Sources dynamiques symboliques markoviennes

Jusqu'à maintenant, les applications de décalage T et de codage σ utilisées à chaque étape sont toujours les mêmes. Très souvent, la modélisation de sources plus proches de la réalité conduit à introduire une nouvelle dépendance : l'application de décalage dépend elle-même du dernier

symbole émis. Cela conduit à considérer des sources nommées «markoviennes» dont les chaînes de Markov classiques sont un cas particulier.
Dans cette section, on reprend l'analogie avec un automate. Le système évolue d'un état à l'autre (l'état est repéré par le symbole obtenu au moyen du codage σ). La différence avec les sources dynamiques de base réside dans le fait qu'à chaque changement d'état, on change également de mécanisme de source (codage et décalage). Ici, nous allons considérer uniquement un alphabet fini.

DÉFINITION 4.2 (SOURCES DYNAMIQUES SYMBOLIQUES MARKOVIENNES). *Soit $\mathcal{M}$ un alphabet fini de cardinal r, et $\mathsf{S} = (\mathsf{S}_0, \mathsf{S}_1, \mathsf{S}_2, \ldots, \mathsf{S}_i, \ldots, \mathsf{S}_r)$ un ensemble de $r+1$ systèmes dynamiques basiques, tous définis sur le même alphabet $\mathcal{M}$. Pour commencer le processus, on utilise le système dynamique S_0. Ensuite, au cours du processus, l'émission d'un symbole j entraîne l'utilisation du système dynamique S_j.*

Nous décrivons de façon plus précise le mécanisme de la source. Pour tout réel x de $\mathcal{I}$, on associe un mot infini $M(x)$ sur l'alphabet $\mathcal{M}$, de même que dans (4.1)

$$M(x) := (M_1(x), \ldots, M_k(x), \ldots),$$

tout en considérant la suite des itérés de x

$$(T^{\langle 1\rangle}(x), T^{\langle 2\rangle}(x), \ldots, T^{\langle k\rangle}(x), \ldots),$$

qui sont définis par la condition initiale $M_1(x) := \sigma_0(x), \;\; T^{\langle 1\rangle}(x) := T_0(x)$, et la relation de récurrence

$$\text{si } M_k(x) = j, \text{ alors } T^{\langle k+1\rangle}(x) := T_j(T^{\langle k\rangle}(x)), \text{ et } M_{k+1}(x) := \sigma_j(T^{\langle k\rangle}(x)). \tag{4.3}$$

Chaque décalage T_j est alors associé à une partition topologique $(\mathcal{I}_{i|j})$, $(1 \le i \le r)$ de l'intervalle $\mathcal{I} = [0,1]$, et satisfait les hypothèses (d) de la définition 4.1 (la condition (d_3) puisque l'alphabet est fini est trivialement vraie). Nous désignons par $(T_{[i|j]})$ la i^{e} branche de T_j. Les applications $T_{[i|j]}$ sont alors des applications réelles expansives analytiques bijectives de $\mathcal{I}_{i|j}$ vers $\mathcal{I}$. Les branches inverses de T_j sont notées $h_{i|j}$. Par définition, $h_{i|j}$ est une application réelle analytique bijective de $\mathcal{I}$ vers $\mathcal{I}_{i|j}$ qui admet une extension analytique sur $\mathcal{V}$, envoyant strictement $\mathcal{V}$ dans $\mathcal{V}$.

Le modèle classique des chaînes de Markov d'ordre 1 est alors le cas ou toutes les sources S_j sont sans mémoire. Plus précisément, si le système S_j est une source sans mémoire de paramètres $\Pi_j := (p_{i|j})_{1\le i\le r}$, la matrice de transition Π de la chaîne de Markov est la matrice $r \times r$

$$\Pi := (p_{i|j}) \quad 1 \le i, j \le r,$$

et le vecteur initial de probabilités n'est autre que le vecteur Π_0.

Relation entre sources markoviennes et systèmes dynamiques généraux. Toute source markovienne peut être associée à un système dynamique défini par un seul décalage qui n'est plus basique. Considérons $r+1$ copies de $\mathcal{I}$, par exemple $\mathcal{I}_0 := \mathcal{I} =]0,1[$ et $\mathcal{I}_j :=]j, j+1[$. Partant d'une source markovienne à r états, on en crée une autre à $r(r+1)$ états. Soit Φ_m la translation $\Phi_m(x) := x + m$. Nous définissons alors, pour $1 \le i \le r$ et $0 \le j \le r$

$$\mathcal{I}_{i,j} := \Phi_j(\mathcal{I}_{i|j}), \qquad T_{i,j} := \Phi_i \circ T_{[i|j]} \circ \Phi_j^{-1},$$

afin que $T_{i,j}$ soit une bijection de $\mathcal{I}_{i,j}$ vers $\mathcal{I}_i$. Le système S associé à la partition $\mathcal{I}_{i,j}$ de $]0, r+1[$ avec les branches $T_{i,j}$ est un système dynamique général.

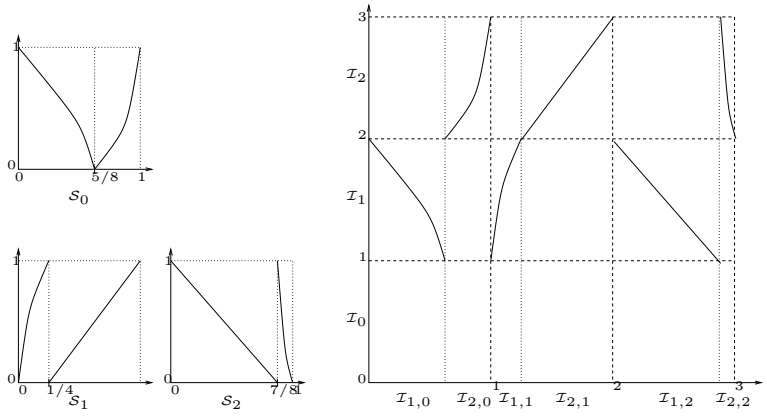


FIG. 4.7 – Relation entre une source dynamique markovienne (définie grâce à trois sources dynamiques de base S_0, S_1, S_2) et le système dynamique général correspondant.

On peut utiliser les deux interprétations d'une source markovienne. Ici, nous préférons rester sur l'intervalle $[0, 1]$ et nous nous en tiendrons au formalisme du paragraphe précédent, plus proche de la structure sous-jacente des chaînes de Markov. C'est le point de vue qui a été choisi par Ruelle lui-même dans [86].

4.3 Sources dynamiques probabilisées

Dans la suite, nous nous intéresserons aux *sources dynamiques probabilisées*, où les mots sont émis à l'aide du mécanisme de source précédemment décrit, mais en permettant une distribution initiale sur $\mathcal{I} = [0, 1]$ déterminée par une fonction de densité. Ajouter une fonction de densité sur $[0, 1]$ permet d'élargir encore le modèle. Cette extension s'inscrit dans le prolongement de travaux Luc Devroye par exemple [21, 22].

DÉFINITION 4.3 (SOURCES DYNAMIQUES PROBABILISÉES). *Soit* S *une source dynamique (basique ou markovienne) et* f *une fonction de densité sur* $\mathcal{I}$ *qui admet une extension analytique sur* $\mathcal{V}$*. Soit* $F(x) = \int_0^x f(t)\,\mathrm{d}t$ *la fonction de distribution associée. La paire* (S, F) *est appelée source dynamique probabilisée. L'ensemble* $\mathcal{M}^{\mathbb{N}}$ *des mots produits par la source dynamique probabilisée* (S, F) *est l'ensemble des mots* $M(\mathcal{I})$ *muni de la probabilité induite par* M *de* f*.*

Explicitons cette notion de mesure de probabilité induite par M de f. La fonction de densité f permet de définir sur $[0, 1]$ une mesure ν sur l'ensemble des Boréliens de $[0, 1]$. Par exemple, la mesure de l'intervalle $[a, b]$ est $\nu([a, b]) = \int_a^b f(t)\mathrm{d}t$. La mesure d'un ensemble de mots $\mathcal{N} \subset \mathcal{M}^{\mathbb{N}}$ est

$$\mu(\mathcal{N}) = \nu(M^{-1}(\mathcal{N})) = \Pr\{M^{-1}(\mathcal{N})\}. \tag{4.4}$$

Cette mesure est bien définie sur les ensembles de mots partageant le même préfixe. Ce sont les boréliens de $\mathcal{M}^{\mathbb{N}}$ définis par

$$\mathcal{N}(\boldsymbol{w}) = \{M(x) \mid \boldsymbol{w} \text{ est préfixe de } M(x)\}.$$

4.4 Intervalles fondamentaux et préfixes

Nous considérons maintenant le k^{e} itéré du décalage. Dans le cas d'une source basique, cela correspond simplement au k^{e} itéré de T pris dans le sens usuel. Dans le cas d'une source markovienne, c'est le décalage $T^{\langle k \rangle}$ au sens de la définition (4.3). Chaque branche (ou branche inverse) du k^{e} itéré du décalage est appelée branche (ou branche inverse) de profondeur k. La profondeur de la branche inverse h est notée $|h|$. Une branche (inverse ou non) de profondeur k est alors associée de façon unique à un mot fini (préfixe) $\boldsymbol{w} = (m_1, m_2, \ldots, m_k)$ de longueur k qui garde la trace des choix effectués jusqu'à présent. Dans le cas basique, chaque branche et chaque branche inverse de profondeur k associée à un préfixe $\boldsymbol{w} = (m_1, m_2, \ldots, m_k)$ est de la forme

$$T_{[\boldsymbol{w}]} = T_{[m_k]} \circ T_{[m_{k-1}]} \circ \cdots \circ T_{[m_1]}, \quad h_{\boldsymbol{w}} = h_{m_1} \circ h_{m_2} \circ \cdots \circ h_{m_k}, \tag{4.5}$$

où $T_{[i]}$ et h_i désignent la i^{e} branche ou la i^{e} branche inverse de T. Dans le cas markovien, chaque branche et chaque branche inverse de profondeur k associé au préfixe $\boldsymbol{w} = (m_1, m_2, \ldots, m_k)$ est de la forme

$$T_{[\boldsymbol{w}]} = T_{[m_k|m_{k-1}]} \circ T_{[m_{k-1}|m_{k-2}]} \circ \cdots \circ T_{[m_1|0]}, \quad h_{\boldsymbol{w}} = h_{m_1|0} \circ h_{m_2|m_1} \circ \cdots \circ h_{m_k|m_{k-1}}. \tag{4.6}$$

Pour un alphabet fini de cardinal r, il y a r^k branches de profondeur k. Les branches cycliques, i.e., les branches pour lesquelles les préfixes associés commencent et finissent par le même symbole, jouent un rôle important dans le cas des sources markoviennes. On désigne l'ensemble des préfixes qui commencent et terminent par le symbole i par $\mathcal{C}[i]$. On note également $\mathcal{C} = \cup_{i \in \mathcal{M}} \mathcal{C}[i]$.

Nous présentons maintenant les objets principaux de cette thèse.

DÉFINITION 4.4 (INTERVALLES FONDAMENTAUX). *L'*intervalle fondamental *associé au préfixe $\boldsymbol{w}$ est l'image $\mathcal{I}_{\boldsymbol{w}} := h_{\boldsymbol{w}}(\mathcal{I})$ de l'intervalle $\mathcal{I} = [0, 1]$ par la branche inverse $h_{\boldsymbol{w}}$. De façon équivalente, l'intervalle fondamental $\mathcal{I}_{\boldsymbol{w}}$ est l'ensemble des réels $M^{-1}(\mathcal{N}(\boldsymbol{w}))$, où $\mathcal{N}(\boldsymbol{w})$ est l'ensemble des mots de $\mathcal{M}^{\mathbb{N}}$ partageant le préfixe $\boldsymbol{w}$ défini en (4.4). La profondeur de l'intervalle fondamental est la longueur $|\boldsymbol{w}|$ du préfixe $|\boldsymbol{w}|$. Les intervalles fondamentaux de profondeur* 1 *sont donc exactement les intervalles de la partition topologique initiale.*

Dans ce contexte, la mesure $u_{\boldsymbol{w}}$ de l'intervalle fondamental $\mathcal{I}_{\boldsymbol{w}}$

$$u_{\boldsymbol{w}} := \mu(\mathcal{N}) = |F(h_{\boldsymbol{w}}(0)) - F(h_{\boldsymbol{w}}(1))|. \tag{4.7}$$

Cette quantité joue un rôle particulièrement important, puisqu'elle est égale à la probabilité que la source produise un mot de $\mathcal{M}^{\mathbb{N}}$ qui commence par le préfixe $\boldsymbol{w}$. Cette mesure $u_{\boldsymbol{w}}$ est ainsi appelée *mesure fondamentale* relative à $\boldsymbol{w}$.

Afin de mieux visualiser géométriquement les intervalles fondamentaux, on dessine plutôt les cercles fondamentaux dont les diamètres coïncident avec les intervalles fondamentaux. La figure 4.8 montre la représentation des intervalles fondamentaux pour le cas de la source sans mémoire du développement binaire, et de la source en fraction continue.

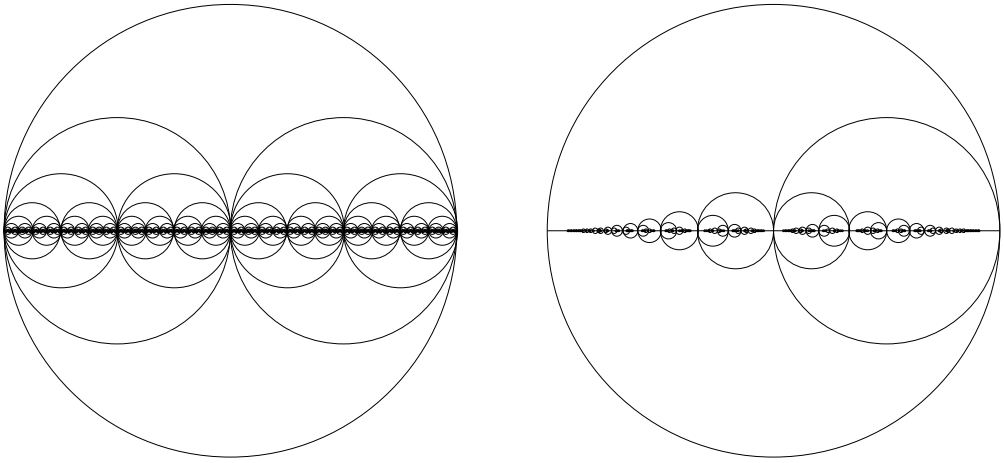


FIG. 4.8 – Les cercles fondamentaux sont représentés pour les sources correspondant au développement binaire (à gauche) et au développement en fraction continue (à droite).

4.5 Série de Dirichlet d'intervalles fondamentaux, entropie et probabilité de coïncidence

Pour une source dynamique probabilisée (S, F), l'*entropie* $h(\mathsf{S}, F)$ est définie comme la limite, si elle existe, d'une quantité faisant intervenir les mesures fondamentales $u_{\boldsymbol{w}}$,

$$h(\mathsf{S}, F) := \lim_{k\to\infty} \frac{-1}{k} \sum_{|\boldsymbol{w}|=k} u_{\boldsymbol{w}} \log u_{\boldsymbol{w}}. \tag{4.8}$$

La probabilité que deux mots émis indépendamment par la même source aient le même préfixe de longueur k est égale à $\sum_{|\boldsymbol{w}|=k} u_{\boldsymbol{w}}^2$. Généralement, au moins pour les sources classiques, cette quantité admet une décroissance exponentielle avec k, ce qui nous amène à définir la *probabilité de coïncidence* $c(\mathsf{S}, F)$ comme le taux de décroissance exponentielle correspondant

$$c(\mathsf{S}, F) := \lim_{k\to\infty} \Big(\sum_{|\boldsymbol{w}|=k} u_{\boldsymbol{w}}^2 \Big)^{1/k}, \tag{4.9}$$

pourvu qu'il existe. Les deux notions d'entropie et de probabilité de coïncidence peuvent s'exprimer à l'aide de la *série des intervalles fondamentaux de profondeur* k

$$\Lambda_k(F, s) := \sum_{|\boldsymbol{w}|=k} u_{\boldsymbol{w}}^s = \sum_{|\boldsymbol{w}|=k} |F(h_{\boldsymbol{w}}(0)) - F(h_{\boldsymbol{w}}(1))|^s. \tag{4.10}$$

On a

$$h(\mathsf{S}, F) := \lim_{k\to\infty} \frac{-1}{k} \frac{d}{ds} \Lambda_k(F, s)\big|_{s=1}, \qquad c(\mathsf{S}, F) := \lim_{k\to\infty} [\Lambda_k(F, 2)]^{1/k}. \tag{4.11}$$

La suite montrera que les quantités $\Lambda_k(F, s)$ définies dans (4.10) se «comportent» asymptotiquement comme des puissances k^{e} d'une certaine fonction $\lambda(s)$ analytique près de l'axe réel

(Proposition 7.14)

$$\Lambda_k(F, s) \approx \lambda(s)^k. \tag{4.12}$$

L'entropie et la probabilité de coïncidence dépendent alors seulement du mécanisme de la source S et non de la distribution F. Elles s'expriment à l'aide de la fonction $\lambda(s)$ et les relations (4.11) et (4.12) permettent de montrer que

$$h(\mathsf{S}) = -\lambda'(1), \qquad c(\mathsf{S}) = \lambda(2).$$

Ces deux grandeurs caractéristiques de la source jouent un rôle très important dans la suite.

Chapitre 5

Tries généraux

Sommaire

Nous commençons par adapter la définition de trie dans le contexte des sources dynamiques. Puis nous proposons une étude en moyenne des paramètres d'un trie standard qui met en avant le lien très fort entre la structure de trie et les intervalles fondamentaux. Si l'on considère ensuite les différentes façons d'implanter le trie, on est amené à introduire la notion de trie hybride. Du point de vue de la complexité, plusieurs paramètres sont intéressants dans cette structure qui peuvent être interprétés directement d'un point de vue algorithmique. Enfin, la structure de ternary search trie, qui résulte de l'hybridation d'un trie avec des ABR, est présentée plus en détail.

5.1 Structure de trie et sources dynamiques

5.1.1 Définition de la structure de trie

La structure de trie introduite au chapitre 2 est ici adaptée de la définition 2.2 dans le cadre de l'analyse.

Soit X une suite de n réels de $[0,1]$, $X = (x_1, \ldots, x_n) \in \mathcal{I}^n$. On considère la suite de chaînes $M(X)$ produites par la source dynamique S,

$$M(X) := (M(x_1), M(x_2), \ldots, M(x_n)).$$

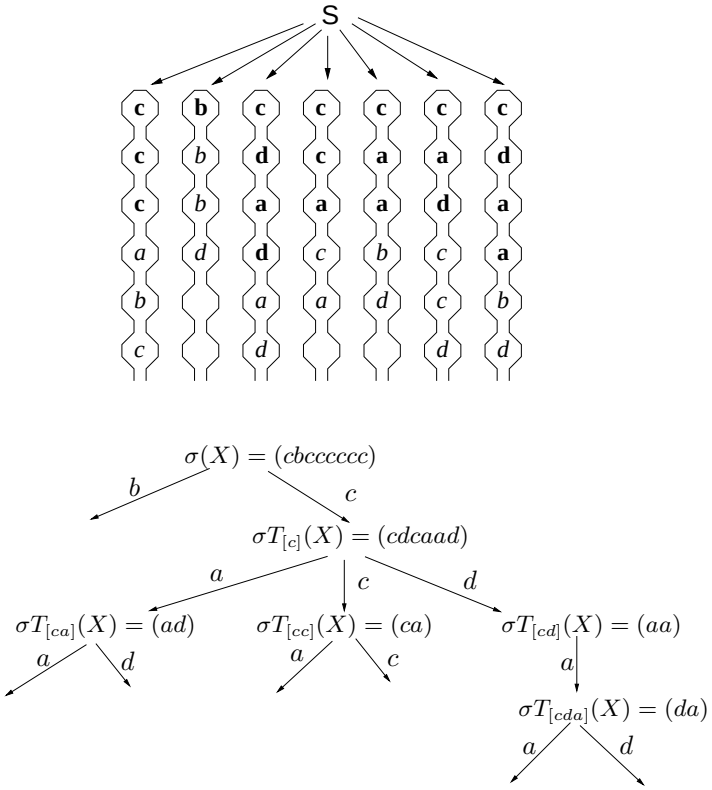


FIG. 5.1 – Une source dynamique S émet indépendamment des chaînes infinies qui sont représentées verticalement (en haut). Par contre, en chaque nœud interne du trie, seules les «tranches», verticales, doivent être considérées.

Nous considérons également la suite $\sigma(X)$ formée par les premiers symboles $\sigma(x_i)$ de chaque chaîne $M(x_i)$. Cette suite de lettres forme un mot que nous appellerons «tranche[1]», et elle s'écrit

$$\sigma(X) := (\sigma(x_1), \sigma(x_2), \ldots, \sigma(x_n)).$$

Ainsi, les symboles des chaînes produites par la source sont considérés de deux points de vue différents : ces symboles forment les mots produits par la source (et sont représentés comme des mots verticaux sur la figure 5.1), mais aussi des mots finis (qui sont représentés comme des tranches horizontales sur la figure 5.1).

La définition 2.2 se transpose directement en adaptant les définitions des opérateurs de décalage $\underline{T}$ et de codage $\underline{\sigma}$ afin de considérer plutôt les opérateurs T et σ définis pour les sources dynamiques dans le chapitre 4. Afin de construire la structure, nous partons de la racine. Tout d'abord, on groupe toutes les chaînes qui commencent par le même symbole m dans une branche étiquetée m, si bien que le sous-arbre correspondant à cette branche regroupe tous les mots commençant

[1]Si l'on voit les chaînes produites comme des rubans sur lesquels se trouvent des lettres, effectivement on rassemble les symboles obtenus en «tranchant» tous les rubans au niveau de la première lettre

par m privés de leur première lettre[2]. Dans notre modèle, supprimer la première lettre d'un mot $M(x)$ est équivalent à considérer le mot relatif au décalé $T(x)$. Dans ce cas, les suffixes de ces chaînes sont associés à la suite des décalés de X

$$T_{[m]}(X) := (T_{[m]}(x_1), T_{[m]}(x_2), \ldots, T_{[m]}(x_n)),$$

où $T_{[m]}(x)$ est défini seulement si le premier symbole $\sigma(x)$ est m (et vaut alors $T(x)$). Ce processus de partitionnement continue jusqu'à ce que tous les mots aient été séparés les uns des autres.

Dans le cadre de l'analyse, nous sommes intéressés par le processus de construction du trie. C'est pourquoi dans cette partie, X n'est plus un ensemble (comme dans le chapitre 2) mais une suite de réels où l'ordre importe. *Les mots $M(x)$ sont insérés dans l'ordre dans le trie.* Afin de repérer localement en chaque nœud interne l'ordre dans lequel les symboles sont «arrivés». On associe à chaque nœud interne étiqueté $\boldsymbol{w}$ la tranche $\sigma(T_{[\boldsymbol{w}]}X)$ (voir figure 5.1).

En résumé, on obtient la définition équivalente à la définition 2.2 mais dans un nouveau cadre

DÉFINITION 5.1 (TRIE DANS LE CADRE DES SOURCES DYNAMIQUES). *Soit X une suite de réels de $[0,1]$. On construit le trie associé à X, noté* $\mathsf{trie}(X)$ *grâce aux règles récursives suivantes :*
- *si* $|X| = 0$, *alors* $\mathsf{trie}(X) = \varnothing$ *est* vide.
- *si* $|X| = 1$ *(i.e.* $X = (x)$*), alors* $\mathsf{trie}(X)$ *est un* nœud externe *étiqueté par* x.
- *si* $|X| \geq 2$, *le trie* $\mathsf{trie}(X)$ *est*

$$\mathsf{trie}(X) = \left\langle \sigma(X), \left\{\mathsf{trie}(T_{[m]}(X))\right\}_{m\in\mathcal{M}} \right\rangle. \tag{5.1}$$

où la tranche $\sigma(X)$ est associée au nœud interne crée et $T_{[m]}(X)$ regroupe les décalés de X donc le premier symbole obtenu par σ est m.

On associe à chaque nœud interne étiqueté par $\boldsymbol{w}$ (correspondant à un chemin d'étiquette $\boldsymbol{w}$), un intervalle fondamental $\mathcal{I}_{\boldsymbol{w}}$. Inversement, pour tout mot $\boldsymbol{w}$ de $\mathcal{M}^*$, le nœud interne d'étiquette $\boldsymbol{w}$ existe si et seulement si l'intervalle fondamental $\mathcal{I}_{\boldsymbol{w}}$ contient au moins deux points de X. Il est ainsi possible d'aborder la structure de trie du point de vue des intervalles fondamentaux.

5.1.2 Approche avec les intervalles fondamentaux

Formalisons un peu le lien unissant la structure de trie aux intervalles fondamentaux. À cette fin, nous effectuons dans cette partie l'analyse en moyenne des paramètres des tries classiques déjà abordés au chapitre 2 (longueur de cheminement, nombre de nœuds internes et hauteur). La notion d'intervalle fondamental apparaît naturellement au cours de l'analyse et permet d'exprimer les espérances des paramètres du trie.
On se place dans le cadre de la définition 5.1 avec un ensemble $X = \{X_1, \ldots, X_n\}$ de n variables aléatoires de même loi tirées dans l'intervalle $[0,1]$ avec une fonction de densité f.

On calcule la probabilité que k points parmi n tombent dans un intervalle fondamental $\mathcal{I}_{\boldsymbol{w}}$. La probabilité de tirer un réel dans $\mathcal{I}_{\boldsymbol{w}}$ est exactement la mesure fondamentale $u_{\boldsymbol{w}}$. Par conséquent la probabilité de l'événement $|X \cap \mathcal{I}_{\boldsymbol{w}}| = k$ est

$$\Pr\{|X \cap \mathcal{I}_{\boldsymbol{w}}| = k\} = \binom{n}{k} u_{\boldsymbol{w}}^k (1 - u_{\boldsymbol{w}})^{n-k}.$$

[2] On remarque que les réels associés aux mots dont le premier symbole est m appartiennent au même intervalle fondamental $\mathcal{I}_m$.

Nombre de nœuds internes

Ainsi le fait qu'un nœud du trie $\mathsf{trie}(X)$ associé à un préfixe $\boldsymbol{w}$ soit interne est équivalent à l'événement $|X \cap \mathcal{I}_{\boldsymbol{w}}| \geq 2$, dont la probabilité est

$$\Pr\{|X \cap \mathcal{I}_{\boldsymbol{w}}| \geq 2\} = \sum_{k=2}^{n} \binom{n}{k} u_{\boldsymbol{w}}^k (1-u_{\boldsymbol{w}})^{n-k} = 1 - (1-u_{\boldsymbol{w}})^n - n u_{\boldsymbol{w}} (1-u_{\boldsymbol{w}})^{n-1}.$$

Le nombre de nœud internes pour un trie construit sur n réels $\{x_1, \ldots, x_n\}$ s'écrit encore sous la forme d'une somme

$$\sum_{\boldsymbol{w} \in \mathcal{M}^*} \delta_{\boldsymbol{w}}(x_1, \ldots, x_n),$$

où la fonction $\delta_{\boldsymbol{w}}(x_1, \ldots, x_n)$ est une fonction indicatrice qui vaut 1 si le nœud associé à $\boldsymbol{w}$ dans le trie construit sur $\{x_1, \ldots, x_n\}$ est interne et 0 sinon. La valeur moyenne du nombre de nœuds internes s'exprime alors grâce à la somme

$$\sum_{\boldsymbol{w} \in \mathcal{M}^*} \Pr\{|X \cap \mathcal{I}_{\boldsymbol{w}} \geq 2|\},$$

en considérant la variable aléatoire $X = \{X_1, \ldots, X_n\}$. Finalement, on obtient donc l'expression T_n de l'espérance du nombre de nœuds interne en fonction des intervalles fondamentaux

$$T_n = \sum_{\boldsymbol{w} \in \mathcal{M}^*} (1 - (1-u_{\boldsymbol{w}})^n - n u_{\boldsymbol{w}} (1-u_{\boldsymbol{w}})^{n-1}).$$

Longueur de cheminement

La longueur de cheminement (externe) est la somme des longueurs des chemins pour aller de la racine à chacun des nœuds externes. On considère toujours $X = (X_1, \ldots, X_n)$ un vecteur de n variables de même loi avec une densité f sur $[0,1]$. Soit $D_{i,n}$ la variable aléatoire donnant la profondeur du nœud externe associé au mot $M(X_i)$ dans le trie $\mathsf{trie}(X_1, \ldots, X_n)$. L'événement $\{D_{i,n} > \ell\}$ admet pour probabilité la probabilité conditionnelle que l'intervalle fondamental $\mathcal{I}_{\boldsymbol{w}}$ de profondeur ℓ ($|\boldsymbol{w}| = \ell$) contienne au moins un des X_j ($j \neq i$) sachant que X_i se trouve dans $\mathcal{I}_{\boldsymbol{w}}$. Les variables X_i sont indépendantes ; on évalue facilement cette probabilité

$$\sum_{k=1}^{n-1} \binom{n-1}{k} u_{\boldsymbol{w}}^k (1-u_{\boldsymbol{w}})^{n-1-k}.$$

La probabilité de l'événement $\{D_{i,n} > \ell\}$ est donc

$$\Pr\{D_{i,n} > \ell\} = \sum_{\boldsymbol{w} \in \mathcal{M}^\ell} u_{\boldsymbol{w}} \sum_{k=1}^{n-1} \binom{n-1}{k} u_{\boldsymbol{w}}^k (1-u_{\boldsymbol{w}})^{n-1-k}.$$

Donc la valeur moyenne de la profondeur moyenne d'un nœud externe est

$$\mathrm{E}[D_{i,n}] = \sum_{\ell=0}^{\infty} \ell \Pr\{D_{i,n} = \ell\} = \sum_{\ell=0}^{\infty} \Pr\{D_{i,n} > \ell\} = \sum_{\boldsymbol{w} \in \mathcal{M}^*} u_{\boldsymbol{w}} [1 - (1 - n u_{\boldsymbol{w}})^{n-1}].$$

L'espérance de la longueur de cheminement est

$$L_n = \sum_{i=1}^{n} \mathrm{E}[D_{i,n}] = n \sum_{\boldsymbol{w} \in \mathcal{M}^*} u_{\boldsymbol{w}} [1 - (1 - n u_{\boldsymbol{w}})^{n-1}].$$

Hauteur

On commence par évaluer, toujours avec le même modèle aléatoire, la probabilité π_ℓ de l'événement «le trie est de hauteur inférieure ou égale à ℓ». La probabilité de cet événement est celle que tous les intervalles fondamentaux de profondeur ℓ contiennent au plus un point. C'est l'événement

$$\prod_{\boldsymbol{w}\in\mathcal{M}^\ell} \{|\mathcal{I}_{\boldsymbol{w}} \cap X| \leq 1\}. \tag{5.2}$$

Nous allons considérer un modèle probabiliste qui permette de considérer que deux événements se rapportant à des intervalles disjoints sont indépendants : le modèle de Poisson. Dans ce modèle, le nombre n de points tirés aléatoirement n'est plus fixé, mais suit une loi de Poisson. Le nombre N de points est donc une variable aléatoire qui suit une loi de Poisson d'un certain paramètre z, c.-à-d.

$$\Pr\{N = k\} = e^{-z}\sum_{k=0}^{\infty}\frac{z^k}{k!}.$$

La moyenne et la variance vérifient alors $\mathrm{E}[N] = \mathrm{V}ar[N] = z$. En choisissant un paramètre $z = n$, du fait de la forte concentration de N autour de z, les deux modèles relatifs à un nombre de points fixé n et un nombre de points suivant une loi de Poisson de paramètre $z = n$ sont assez «proches». Le chapitre 6 donnera une description plus détaillée de ces deux modèles appelés modèles de Bernoulli et de Poisson. Ici, le but est simplement de mettre en évidence le lien entre le calcul des valeurs moyennes des paramètres de trie et les intervalles fondamentaux. Dans le modèle de Poisson, on a une forte propriété d'indépendance entre événements associés à des intervalles disjoints.

LEMME 5.2 (INDÉPENDANCE DU MODÈLE DE POISSON). *Soit une famille finie d'intervalles disjoints $\{\mathcal{J}_i\}_{1\leq i\leq r}$ inclus dans $[0,1]$ ($\cap_{1\leq i\leq r}\mathcal{J}_i = \varnothing$ et $\cup_{1\leq i\leq r}\mathcal{J}_i \subset [0,1]$) et une suite $X = (X_1,\ldots,X_N)$ de variables aléatoires de même loi de fonction de densité f, avec N une variable de loi de Poisson de paramètre z. Alors l'événement $\cap_{1\leq i\leq r}\{|X\cap\mathcal{J}_i = n_i|\}$ a pour probabilité*

$$\prod_{1\leq i\leq r}\Pr\{|X\cap\mathcal{J}_i| = n_i\}.$$

Preuve. En effet, soit u_i la probabilité de tirer un X_i dans $\mathcal{J}_i$, i.e. $u_i = \displaystyle\int_{x\in\mathcal{J}_i} f(x)\,\mathrm{d}x$. La probabilité de l'événement $|X\cap\mathcal{J}_i| = k$ vaut

$$\Pr\{|X\cap\mathcal{J}_i = k|\} = \sum_{n=k}^{\infty}\Pr\{N=n\}\binom{n}{k}u_i^k(1-u_i)^{n-k} = e^{-zu_i}\frac{u_i^k}{k!}.$$

Posons $s = n_1 + n_2 + \cdots + n_r$ et $\overline{u} = 1 - \sum_{i=1}^r u_i$ (la probabilité de tirer un point en dehors de l'union des $\mathcal{J}_i$). Alors, on peut écrire dans le modèle de Poisson de paramètre z

$$\begin{aligned}\Pr\{\cap_{1\leq i\leq r}|X\cap\mathcal{J}_i = n_i|\} &= \sum_{k=n_1+\ldots+n_r}^{\infty}\Pr[N=k]\frac{k!}{n_1!\cdots n_r!(k-s)!}u_1^{n_1}\ldots u_r^{n_r}\overline{u}^{k-s}\\ &= \prod_{i=1}^r e^{-zn_i}\frac{u_i^{n_i}}{n_i!} = \prod_{i=1}^r\Pr\{|X\cap u_i| = n_i\}.\end{aligned}$$

□

En vertu de ce lemme, on calcule la probabilité de l'événement (5.2) ou encore $\{H \leq \ell\}$ (où H est la variable aléatoire associée à la hauteur du trie construit sur X)

$$\Pr\{H \leq \ell\} = \prod_{\boldsymbol{w} \in \mathcal{M}^\ell} \Pr\{|X \cap \mathcal{I}_{\boldsymbol{w}}| \leq 1\} = \prod_{u_{\boldsymbol{w}} \in \mathcal{M}^\ell} (1 - e^{-z u_{\boldsymbol{w}}}).$$

Par conséquent, l'espérance H_z de la hauteur est dans le modèle de Poisson de paramètre z

$$H_z = \sum_{\ell=0}^{\infty} \Pr\{H \leq \ell\}(1 - \Pr\{H \leq \ell\}) = \sum_{\ell=0}^{\infty} (1 - \prod_{u_{\boldsymbol{w}} \in \mathcal{M}^\ell} (1 - e^{-z u_{\boldsymbol{w}}})).$$

On voit ici que le modèle de Poisson est souvent plus simple à manier. Dans un premier temps, on réalise souvent l'analyse dans le modèle de Poisson, avant de revenir au modèle plus naturel de Bernoulli (à taille fixée) grâce une étape de dépoissonisation. Ces méthodes seront présentées aux chapitres 6 et 8.

La structure de trie définie au début de ce chapitre est donc intimement liée aux intervalles fondamentaux. Nous envisageons maintenant des structures hybrides qui tentent de se rapprocher d'une structure donnée «réelle» et dont nous effectuerons l'analyse en moyenne du paramètre de longueur de cheminement.

5.2 Tries hybrides

Si l'on considère la structure abstraite représentant un trie, chaque nœud interne possède un lien vers ses fils. Avec un alphabet totalement ordonné, il existe trois manières naturelles d'implanter un nœud et sa descendance à l'aide des structures de données classiques que sont les tableaux, les listes ordonnées et les arbres binaires de recherche. La structure obtenue constitue un trie «hybride» alliant la structure globale du trie avec celle d'autres structures de données (qui permettent de passer d'un nœud à l'un de ses fils).

(a) L'implantation la plus simple utilise des tableaux dont la taille est le cardinal de l'alphabet. On accède aux fils d'un nœud directement grâce à un tableau de pointeurs. Cette solution n'a donc pas de sens si l'alphabet est infini et reste peu efficace du point de vue de l'occupation mémoire si l'alphabet est assez grand (puisque beaucoup de pointeurs nuls sont alloués). Cette solution est tout à fait adaptée au cas d'un alphabet de petite taille (typiquement pour stocker des mots binaires).

(b) La structure de «trie-liste» remédie au surcoût en termes de place mémoire du trie-tableau en créant des liens entre sous-arbres frères, mais en remplaçant ainsi l'accès direct par un parcours de liste chaînée (ordonnée). Cela revient à représenter le trie en tant qu'arbre général sous forme d'un arbre binaire à l'aide de la transformation «fils gauche/frère droit».

(c) La structure de «trie-ABR» utilise des arbres binaires de recherche (ABR) pour représenter la descendance d'un nœud. L'objectif est d'allier les avantages du trie-tableau pour l'efficacité en temps et les avantages des tries-liste pour le gain de place mémoire. Comme il en est fait mention dans l'introduction, le trie hybride ainsi obtenu est strictement équivalent au TST (*ternary search trie*) proposée récemment dans [8] par Bentley et Sedgewick). En effet, le trie-ABR peut être envisagé comme un arbre ternaire dans lequel la recherche des symboles est faite de façon standard dans un arbre binaire de recherche (les liens droits sur la figure 5.2(iii)) sur l'alphabet $\mathcal{M}$, tandis que la descente dans la structure de trie est effectuée à l'aide d'un pointeur d'échappement (liens courbes sur la figure 5.2(iii)) dès que l'égalité des symboles est détectée.

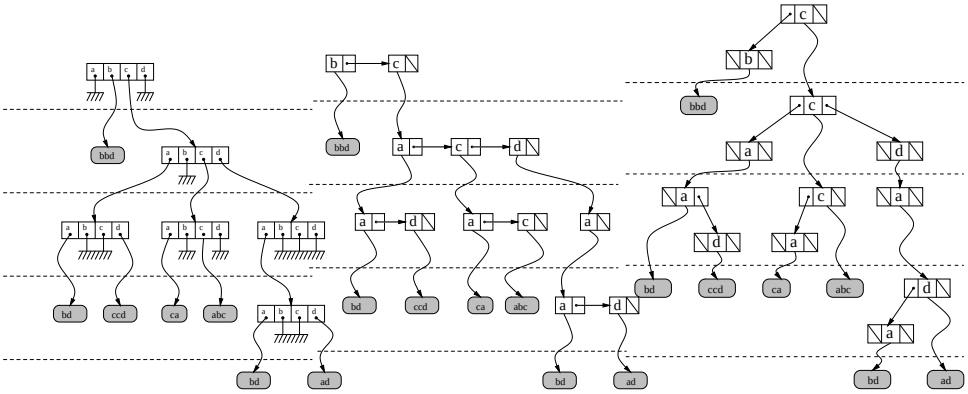


FIG. 5.2 – Trois représentations d'un trie formé sur la suite de mots $(\mathbf{ccc}abc, \mathbf{b}bbd, \mathbf{cdad}ad, \mathbf{cca}ca, \mathbf{caa}bd, \mathbf{cad}ccd, \mathbf{cdaa}bd)$ sur l'alphabet $\mathcal{A} = \{a, b, c, d\}$ (les pointeurs nuls sont représentés comme reliés à la terre).

5.2.1 Paramètres

Les paramètres des tries hybrides sont évidemment reliés aux paramètres de trie classique (voir la partie 2.1.4).

Longueur de cheminement

Dans un trie hybride, la longueur de cheminement et la somme de deux quantités. La première est relative à la structure de trie sous-jacente. La deuxième est le coût additionnel occasionné par la traversée de structures de données aux nœuds internes. Pour calculer la longueur de cheminement globale, on ajoute la longueur de cheminement du trie abstrait et la longueur de cheminement additionnelle. C'est ce surcoût qui est analysé pour les tries hybrides. On évalue dans le cadre de l'analyse en moyenne l'impact de l'hybridation de la structure de trie.

Hauteur

La hauteur d'un trie hybride constitue également un paramètre intéressant, mais non décomposable comme pour la longueur de cheminement. L'étude en moyenne de la hauteur d'un trie hybride reste encore un problème ouvert à ce jour. C'est la longueur de la branche la plus longue en ne différentiant pas les liens de type trie et les liens appartenant aux structures aux nœuds.

Sur la figure 5.2, trois implantations du même trie «abstrait» sont représentés. Pour le trie standard, que ce soit le trie abstrait ou la version trie-tableau, la taille (le nombre de nœud interne) et la longueur de cheminement (la somme des longueurs des chemins pour aller de la racine à chacun des nœuds externes) sont respectivement 6 et 21. Pour les versions hybrides, les pointeurs relatifs aux structures de données «intermédiaires» aux nœuds introduisent un surcoût (en nombre de liens empruntés) pour la longueur de cheminement de 15 pour le trie-liste et 8 pour le trie-ABR.

Remarque. La taille (nombre de nœuds internes) n'est pas un paramètre facile à définir pour les tries hybrides. Pour le trie ABR et le trie-liste, on peut définir la taille comme le nombre de «cel-

lules» (les structures utilisées pour représentées les listes chaînées ou les arbres binaires de recherche). La taille du trie hybride (liste ou ABR) construit sur un ensemble X est

$$\mathsf{size}(\mathsf{trie}(X)) - 1 + |X|,$$

où $\mathsf{size}(\mathsf{trie}(X))$ désigne le nombre de noeuds internes du trie standard et $|X|$ est le nombre d'éléments de X.

5.2.2 Position du problème : paramètres additifs

Un trie classique construit sur une suite de mots correspondant à une suite X d'éléments de $\mathcal{I}^n$ dépend en fait uniquement de l'ensemble sous-jacent à X (c.-à-d. que l'ordre n'a pas d'importance). Cependant, les autres structures de tries (trie-liste et trie-abr) dépendent de l'ordre relatif des éléments de la suite X.

Si l'on se place en un nœud du trie associé au préfixe $\boldsymbol{w}$, dans toutes les implantations, la structure utilisée pour accéder aux nœuds fils est toujours complètement déterminée par la «tranche» (qu'il faut visualiser comme une suite de lettres, donc comme un mot fini)

$$\sigma T_{[\boldsymbol{w}]}(X) := (\sigma T_{[\boldsymbol{w}]}(x_1), \sigma T_{[\boldsymbol{w}]}(x_2), \ldots, \sigma T_{[\boldsymbol{w}]}(x_n)).$$

Du point de vue des intervalles fondamentaux, on construit la structure en un nœud associé au préfixe $\boldsymbol{w}$ en examinant les symboles $\sigma(x_i)$ correspondant aux points x_i qui tombent dans $\mathcal{I}_{\boldsymbol{w}}$, *en conservant leur ordre d'arrivée* (x_i arrive avant x_j si $i < j$).

Remarque – L'application $T_{[\boldsymbol{w}]}$ opère comme une projection ; en effet, appliquée à un ensemble X, on obtient la suite des décalés de X obtenue en considérant seulement les points de X qui ont pour préfixe $\boldsymbol{w}$. Ainsi, dans le cas du développement en base 2, pour un ensemble de points $X = \{x_i\}$ dans $[0, 1[$, l'ensemble $T_{[01]}(X)$ ne contient que les décalés des points x_i dont le développement commence par 01 (c.-à-d. les décalés de l'ensemble $X \cap [\frac{1}{4}, \frac{1}{2}[$).

Soit γ un paramètre «additif» du trie $\mathsf{trie}(X)$ défini récursivement par la relation qui fait intervenir une fonction de coût δ

$$\gamma[\mathsf{trie}(X)] = \delta(\sigma(X)) + \sum_{m \in \mathcal{M}} \gamma[\mathsf{trie}(T_{[m]}(X))].$$

Le paramètre δ, appelé «péage», est un paramètre sur des mots finis. On exige aussi que $\delta(\boldsymbol{s})$ soit nul si $|\boldsymbol{s}| < 2$. Ainsi le paramètre est nul dès lors que le nœud interne n'existe pas (pour la définition du trie).
La relation de récurrence peut alors être déroulée, et on obtient

$$\gamma[\mathsf{trie}(X)] = \sum_{\boldsymbol{w} \in \mathcal{M}^*} \delta[\sigma T_{[\boldsymbol{w}]}(X)], \tag{5.3}$$

(toujours pourvu que $\delta(\boldsymbol{s})$ ait une valeur nulle pour les tranches qui contiennent 0 ou 1 symbole).

Nous décrivons maintenant le modèle probabiliste induit en chaque nœud d'étiquette $\boldsymbol{w}$ du trie par un modèle de Poisson $\mathcal{P}_z$. Le nombre de chaînes insérées dans le trie est lui-même une variable aléatoire de Poisson. Puisque la probabilité qu'un mot infini commence par un préfixe $\boldsymbol{w}$ est égale à la mesure fondamentale $u_{\boldsymbol{w}}$ définie dans (4.7), la probabilité que le prochain symbole émis soit m vaut

$$p_{\boldsymbol{w},m} = \frac{u_{\boldsymbol{w}\cdot m}}{u_{\boldsymbol{w}}}. \tag{5.4}$$

Ici, la notation $\boldsymbol{w}\cdot m$ désigne le mot obtenu par concaténation de $\boldsymbol{w}$ et du symbole m. Puisque tous les éléments de X sont tirés indépendamment, en un nœud interne d'étiquette $\boldsymbol{w}$, les symboles de la tranche $\sigma T_{[\boldsymbol{w}]}X$ apparaissent indépendamment avec des probabilités $\{p_{\boldsymbol{w},m}\}_{m\in\mathcal{M}}$. Il est important de noter que ces tranches sont des mots finis produits par une source sans mémoire de paramètre $\{p_{\boldsymbol{w},m}\}_{m\in\mathcal{M}}$ quelle que soit la source initiale.

De plus, si le cardinal de X est une variable de Poisson de paramètre z, la taille de la tranche $\sigma T_{[\boldsymbol{w}]}(X)$ est également une variable de Poisson de paramètre $zu_{\boldsymbol{w}}$. Il s'ensuit de (5.3) que l'espérance du paramètre γ est une somme d'espérances du paramètre δ,

$$\mathrm{E}[\gamma;\mathcal{P}_z,\mathsf{S},F]=\sum_{\boldsymbol{w}\in\mathcal{M}^*}\mathrm{E}[\delta;\mathcal{P}_{zu_{\boldsymbol{w}}},\{p_{\boldsymbol{w},m}\}], \tag{5.5}$$

où $\boldsymbol{w}$ décrit $\mathcal{M}^*$ l'ensemble des mots (finis), et P_z désigne le modèle de Poisson de paramètre z.

Si le modèle de Poisson apparaît deux fois dans la formule (5.5), il n'a pas la même signification des deux cotés de l'égalité. Le modèle de Poisson du membre de gauche est relatif au nombre de mots infinis émis par une source dynamique probabilisée (la taille est donc le cardinal de l'ensemble des mots). Le modèle de Poisson du membre de droite est relatif à la taille de mots finis (c.-à-d. la longueur), et les mots sont émis par une source sans mémoire.

C'est le deuxième modèle qui fera l'objet du prochain chapitre. Avant de mener à bien l'analyse, nous revenons à un point de vue algorithmique et présentons plus en détail la structure hybride de Ternary Search Trie.

5.3 Ternary search trie

Dans cette section, nous présentons plus en détail la version ABR des tries hybrides, également appelée ternary search trie par Jon Bentley et Robert Sedgewick. Le code C présenté est quasiment le même que celui proposé par les deux auteurs[3]. Cette structure semble profiter du meilleur de deux mondes : l'efficacité en temps des tries et l'efficacité en espace des ABR.

5.3.1 Structure de données et opérations de base

La structure d'un nœud

Il y a deux façons de regarder un ternary search trie : soit comme un trie hybride (c'est le choix qui a été fait pour l'analyse) dont on différencie les liens selon qu'ils appartiennent à la structure globale de trie ou aux structures ABR aux nœuds ; soit comme un arbre ternaire dont les nœuds sont des extensions de nœuds d'un ABR. Examinons le principe de la recherche dans un ABR, la racine contient une clé x et les sous-arbres gauche et droit sont eux-mêmes des ABR contenant respectivement les éléments (strictement) inférieurs à x et (strictement) supérieurs à x (voir la figure 5.3(i)). À la racine d'un TST, le symbole x guide la recherche de la façon suivante : on compare le premier symbole de la chaîne recherchée à x ; de même que dans l'ABR, en cas d'inégalité on poursuit la recherche dans le sous-arbre gauche ou droit ; par contre en cas d'égalité, on passe dans le sous-arbre du milieu (voir la figure 5.3(ii)). En langage C, on a donc la structure du code 5.1. On stocke le symbole x dans la variable `splitchar` et les trois pointeurs représentent les trois enfants du nœud. La racine est déclarée `Tptr *root`. La structure de TST originelle de Bentley et Sedgewick code l'ensemble des chaînes (y compris puisqu'on a choisi la représentation en C un caractère nul `'0'` de terminaison de chaque chaîne). Dans un souci d'efficacité et

[3] J'ai essayé d'éviter d'utiliser les structures C trop spécifiques comme `(a?b :c);` ou l'emploi systématique de la valeur 0 pour faux. Le but est de donner une description algorithmique des différentes fonctionnalités offertes par le TST en présentant un code C qui fonctionne et reste proche de la «vision» algorithmique (voir [8]).

(i) Principe binaire de recherche (ii) Principe ternaire de recherche

FIG. 5.3 – Les deux principes de recherche utilisés dans les arbres binaires de recherche et les ternary search tries.

```
typedef struct tnode *Tptr;
typedef struct tnode {
   char splitchar;
   Tptr lokid, eqkid, hikid;
} Tnode;
```

Code 5.1: Structure en langage C pour représenter un nœud.

d'uniformité au niveau de la représentation, c'est donc en fait un arbre digital plutôt qu'un trie qu'on utilise.

Principe de la recherche, insertion, suppression

Nous commençons par le principe de recherche qui reprend le principe (ii) de la figure 5.3 récursivement. Le code C de la fonction récursive de recherche `rsearch` est présenté en 5.2. Cette fonction renvoie 1 ou 0 selon que le mot est présent ou non dans l'arbre.

L'insertion utilise le même type de parcours dans l'arbre. Une fois encore nous remarquons que le TST développe pour un mot la branche complète correspondante (code 5.3).

La suppression supprime les nœuds de la branche «filaire» sur le chemin qui mène de la racine au nœud externe (code 5.4). Lorsqu'on doit supprimer un nœud on prend garde de réorganiser l'ABR représentant les enfants du nœud. Cela se fait classiquement grâce à la fonction du code 5.5.

Le code présenté ici sert essentiellement à présenter la structure de donnée et montrer que le code est très court (même s'il peut être subtil). L'article de Bentley et Sedgewick [8] (voir également [90]) présente bon nombre d'optimisations possibles. Les versions itératives peuvent être

```
int rsearch(Tptr p, char *s) {
   if (p == NULL) return 0;
   if (*s< p->splitchar)
      return rsearch(p->lokid, s);
   else if (*s > p->splitchar)
      return rsearch(p->hikid, s);
   else {
      if (*s == '\0') return 1;
      return rsearch(p->eqkid, ++s);
   }
}
```

Code 5.2: Recherche dans un TST.

```
Tptr rinsert(Tptr p, char *s) {
   if (p == NULL) {
      p = (Tptr) malloc(sizeof(Tnode));
      p->splitchar = *s;
      p->lokid = p->eqkid = p->hikid = NULL;
   }
   if (*s < p->splitchar)
      p->lokid = rinsert(p->lokid, s);
   else if (*s == p->splitchar) {
      if (*s != 0)
         p->eqkid = rinsert(p->eqkid, ++s);
   } else
     p->hikid = rinsert(p->hikid, s);
   return p;
}
```

Code 5.3: Insertion dans un TST.

```
Tptr rdelete(Tptr p, char *s) {
   Tptr q=p;
   if (p == NULL) return NULL;
   if (*s < p->splitchar)
      p->lokid = rdelete(p->lokid, s);
   else if (*s > p->splitchar)
      p->hikid = rdelete(p->hikid, s);
   else {
      if (*s == '\0') {
         if (p->splitchar != '\0')
            return p; /* string not present */
      } else
         p->eqkid = rdelete(p->eqkid, ++s);
      if (p->eqkid == NULL) return deleteRoot(p);
   }
   if (p->eqkid == NULL) return deleteRoot(p);
}
```

Code 5.4: Suppression dans un TST.

```
Tptr deleteRoot(Tptr p) {
   Tptr q,r;
   q=p;
   if (p->lokid == NULL)
      r = p->hikid;
   else if (p->lokid) {
      r = p->lokid;
      r->hikid=p->hikid;
      q = p;
      while (q->hikid) q=q->hikid;
      q->hikid = p->hikid;
   }
   free(p);
   return r;
}
```

Code 5.5: Suppression d'un nœud du TST menant à une branche filaire.

sensiblement plus efficaces, mais surtout on passe beaucoup de temps à allouer de chaque nœud en mémoire. On peut améliorer sensiblement l'efficacité du code grâce à une gestion de la mémoire plus complexe (allocation par blocs).

La fonction de suppression d'un mot dans un TST n'est pas très efficace. Mais de fait, le TST est adapté à un ensemble de mots où les insertions sont plus fréquentes que les suppressions.

Pour un TST, l'ordre d'insertion des mots n'est pas anodin. Si l'ordre d'insertion dans un trie abstrait est indifférent (le trie sera toujours le même), il n'en est pas de même pour les arbres binaires de recherche : l'ordre d'insertion est alors crucial. Dans un ABR, insérer les clés dans le bon ordre (c.-à-d. la clé médiane d'abord) permet d'obtenir un arbre équilibré ; par contre, insérer les clés en ordre trié donne un arbre filaire «dégénéré» qui se comportera comme une liste.

Les TST, résultat de l'hybridation de ces structures, se situent entre ces deux extrêmes. On peut construire un arbre «relativement» équilibré en équilibrant les ABR aux nœuds du trie. Une autre technique, plus simple, consiste à partir d'un ensemble de mots trié au préalable, d'insérer le mot médian et à poursuivre récursivement sur les deux sous-ensembles de mots. Cela permet d'équilibrer efficacement les ABR qui se trouvent aux profondeurs les moins élevées. Nous verrons au chapitre 9 que c'est bien à ces faibles profondeurs qu'il convient d'éviter les trop grands déséquilibres dans les ABR.

D'autres techniques permettent d'équilibrer les arbres binaires de recherche de manière dynamique (voir en particulier [93] et les *splay trees*). Enfin, dans le cas d'un dictionnaire statique, les arbres binaires de recherche peuvent être organisés de façon optimale par rapport à des coûts d'accès prévus aux nœuds. On affecte à chaque nœud un poids (correspondant par exemple à des statistiques d'accès aux nœuds), et on réorganise tous les arbres de recherche de façon à avoir pour le TST une longueur de cheminement pondérée minimale (on reviendra sur cet aspect au chapitre 9 dans le cadre du correcteur orthographique `epelle`).

5.3.2 Comparaison avec d'autres structures

Un ensemble de chaînes de caractères est typiquement représenté grâce à une table de hachage. On a déjà vu au chapitre 2, que les tries intervenaient dans le hachage dynamique. Ici les TST apparaissent comme des concurrents directs (et sérieux) des tables de hachage.

Nous reprenons les expérimentations de [8] qui pour représenter n chaînes utilisent pour la table de hachage un tableau de taille `tabsize` $= n$ de listes chaînées. La fonction de hachage, tirée du livre de Kernighan et Ritchie *The C programming language*, est raisonnablement efficace tout en assurant une bonne dispersion des clés (voir le code 5.6).

Afin de comparer les deux techniques (tables de hachage et TST) de manière plus juste, on remplace l'appel à la fonction de comparaison de chaînes `strcmp` de la librairie C par avec un code équivalent. Les fonctions de recherche des tables de hachage et du TST sont alors de même style. Nous reprenons les temps expérimentaux obtenus par Bentley et Sedgewick correspondant aux temps de construction des deux types de structure pour un même dictionnaire (où pour le TST, on insère d'abord la chaîne médiane à partir d'un tableau de chaîne trié et ainsi de suite récursivement). Dans un deuxième temps, chacun des mots du dictionnaire est recherché dans les deux structures. On obtient les temps (en secondes) de la table suivante (pour un dictionnaire `/usr/dict/words` standard) :

Machine	Construction		Recherche (positive)	
	hachage	TST	hachage	TST
UltraSPARC-2	0,53	0,49	0,58	0,52
MIPS R4000	0,83	0,71	0,55	0,64
Pentium Pro	0,93	0,71	0,55	0,64

```
typedef struct hnode *Hptr;
typedef struct hnode {
    char    *str;
    Hptr    next;
} Hnode;

Hptr tab[tabsize];

int hashfunc (char *s) {
   unsigned n=0;
   for ( ; *s; s++)
      n = 31 * n + *s;
      return n % tabsize;
}

int hsearch(char *s) {
   for (p = tab[hashfunc(s)]; p; p = p->next)
      if (strcmp(s, p->str) == 0) /* comparai-
son de la chaîne */
         return 1;
   return 0;
}
```

Code 5.6: Fonction de hachage utilisée et fonction de recherche dans une table de hachage.

Avec un dictionnaire français encodé ISO-latin1 qui a été constitué dans le cadre du correcteur orthographique `epelle` (cf. chapitre 9), on obtient sur d'autres machines[4] les temps suivant

Machine	Construction		Recherche (positive)	
	hachage	TST	hachage	TST
UltraSPARC-2	0,34	0,62	0,42	0,39
Pentium II 266MHz	0,32	0,57	0,50	0,50

Les temps sont tout à fait comparables pour les deux structures. Les TST sont généralement plus rapides pour décider qu'un mot n'est pas dans le dictionnaire (recherche négative). En effet, les TST peuvent différencier rapidement deux chaînes dès les premiers caractères tandis que les tables de hachage nécessitent le calcul de deux clés de hachage et donc la lecture complète des deux chaînes. Pour un ensemble de chaînes assez longues et se différenciant dès les premiers caractères, le gain de temps est conséquent.

Le TST est une structure plus coûteuse en place mémoire qu'une table de hachage. On économiserait de la place mémoire en ne développant pas les branches jusqu'au bout (i.e. en passant de l'arbre digital au trie). Cela nécessite d'ajouter des informations aux nœuds et rend le code un peu moins performant. Dans le cas de dictionnaires statiques, on peut aussi compresser l'arbre (cette technique est utilisée dans `epelle`, chapitre 9).

5.3.3 autres fonctionnalités

De manière générale, la structure de TST peut être utilisée dans toutes les applications où apparaissent les tries (cf. chapitre 2).

[4] Avec l'option `-O3` sur le compilateur `gcc`.

```
void rtraverse(Tptr p, char *s, int i) {
   if (p == NULL) return;
   rtraverse(p->lokid,s,i);
   s[i]=p->splitchar;
   if (p->splitchar!='\0')
      rtraverse(p->eqkid, s, i+1);
   else
      printf("%s\n", s);
   rtraverse(p->hikid, s, i);
}

void rtraversemain(Tptr p) {
  static char word[256];
  rtraverse(p,word,0);
}
```

Code 5.7: Accés séquentiel du TST.

Accès séquentiel et statistiques

Certaines applications nécessitent de parcourir séquentiellement un ensemble de mots. Les tables de hachage ne sont pas du tout adaptées à ce type d'utilisation. Par contre, les TST se prêtent naturellement à un parcours séquentiel de l'ensemble des mots dans l'ordre lexicographique. Par exemple pour afficher tous les mots contenus dans un TST, on peut appeler la fonction récursive du code 5.7. On maintient au fur et à mesure du parcours de l'arbre le préfixe correspondant au nœud courant. La variable `i` mémorise la longueur du préfixe courant.
En rajoutant un champ de comptage en chaque nœud, on peut faire des statistiques. Le fait d'avoir des arbres binaires de recherche autorise les recherches par intervalles. Le prédécesseur ou successeur immédiat d'une chaîne est accessible de manière efficace (ce qui n'est pas le cas des tables de hachage). Les fonctionnalités du trie sont donc disponibles pour les TST pour un coût logarithmique alors que les tables de hachage demandent un coût linéaire.

Recherche partielle (recherche de type «mots croisés»)

Le problème de recherche partielle est facile à appréhender pour qui a déjà fait des mots croisés. On recherche simplement une chaîne pouvant contenir deux types de symboles : les lettres de l'alphabet bien sûr, et le symbole '`.`' signifiant «n'importe quel symbole».
La fonction de recherche partielle du code 5.8 affiche les chaînes trouvées. Les variables `tmp` et `i` sont des variables contenant respectivement la chaîne et sa longueur pour le nœud courant.

Recherche de voisinage

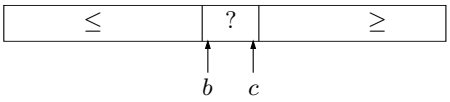
La structure de TST se prête également à la recherche de voisinage : on recherche tous les mots du dictionnaire qui sont à une certaine distance de Hamming d'un mot.
La fonction `nearsearch` réalise cette recherche dans un arbre ternaire de recherche. Elle prend en argument un arbre, une chaîne (la requête) et une distance (voir code 5.9). La recherche est relativement efficace mais les performances se dégradent dès que la distance devient trop grande (puisque la portion de l'arbre visitée augmente rapidement avec la distance passée en argument).
Remarque. Il est utile de modifier la notion de distance pour prendre en compte les permutations de deux lettres consécutives dans le cadre d'un correcteur orthographique. C'est en effet une des fautes de frappe les plus courantes.

```
void pmsearch(Tptr p, char *s, char *tmp, int i) {
   if (p == NULL) return;
   if (*s == '.' || *s < p->splitchar)
      pmsearch(p->lokid, s, tmp, i);
   if (*s == '.' || *s == p->splitchar) {
      tmp[i]=p->splitchar;
      pmsearch(p->eqkid, s+1, tmp, i+1);
   }
   if (*s == '\0' && p->splitchar == '\0')
     printf("*** pm: %s\n",tmp); /* a revoir */
   if (*s == '.' || *s > p->splitchar)
      pmsearch(p->hikid, s, tmp, i);
}

void pmsearchmain(Tptr p, char *s) {
  static char word[256];
  pmsearch(p, s, word, 0);
}
```

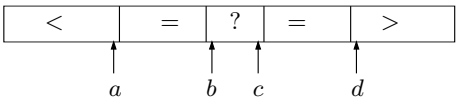
Code 5.8: Recherche partielle

```
void nearsearch(Tptr p, char *s, int d, char *tmp, int i) {
   char *t, e;
   if (p == NULL || d < 0) return;
   if (d > 0 || *s < p->splitchar)
      nearsearch(p->lokid, s, d, tmp, i);
   if (p->splitchar == 0) {
      if ((int) strlen(s) <=d) {
         tmp[i]=p->splitchar;
 printf("*** ns: %s\n",tmp); /* a revoir */
      }
   } else {
      t=s;
      e=d;
      if (*s == '\0')
         t++;
      if (*s != p->splitchar)
         e--;
      tmp[i]=p->splitchar;
      nearsearch(p->eqkid, t, e, tmp, i+1);
   }
   if (d > 0 || *s > p->splitchar)
   nearsearch(p->hikid, s, d, tmp, i);
}

void nearsearchmain(Tptr p, char *s, int d) {
  static char word[256];
  nearsearch(p, s, d, word, 0);
}
```

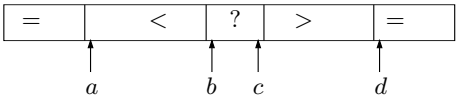
Code 5.9: Recherche de voisinage.



(i) partitionnement classique



(ii) partitionnement «idéal»



(iii) partitionnement avec clés égales

FIG. 5.4 – Le principe de partitionnement naturel (i), le partitionnement «idéal» (ii) et le principe introduit par McIlroy et Bentley (iii).

5.3.4 Algorithme de tri

De la même façon que l'ABR et le tri rapide (quicksort) sont intimement liés, la structure de TST permet de construire une procédure de tri des chaînes de caractères qui s'avère une des plus efficaces. Cette technique de tri se révèle également avantageuse dans certaines situations pour trier l'ensemble des suffixes d'un texte, ce qui survient pour certaines techniques de compression (`bzip2` par exemple l'utilise pour réaliser la transformée de Burrow-Wheeler sur un texte).
L'algorithme de tri lié aux TST utilise le raffinement proposé par Bentley et McIlroy [10] relatif à l'algorithme Quicksort pour des clés égales. (Nous reviendrons également sur ce raffinement dans le cadre des structures nœuds au chapitre 6.)

Le principe de Quicksort consiste à partitionner par rapport à un *pivot*. Idéalement, pour Quicksort, on souhaiterait que toutes les clés égales au pivot se retrouvent en place avec toutes les clés de valeur inférieure à gauche et toutes les clés de valeur supérieure à droite. Il s'agit en fait du *problème du drapeau hollandais* popularisé par Dijkstra [23] : des pierres de couleur rouge, bleue et blanche sont à ordonner dans le même ordre que les couleurs du drapeau hollandais. Cela correspond au partitionnement de Quicksort où les pierres rouges symbolisent les éléments inférieurs au pivot, les pierres blanches les éléments égaux et les pierres bleues les éléments supérieurs. Bentley et McIlroy ont proposé un principe de partitionnement qui bien que peu intuitif s'avère très efficace (voir figure 5.4). La boucle de partitionnement consiste elle-même en deux boucles internes. La première boucle interne incrémente le compteur b : on passe sur les éléments de valeur strictement inférieure, on échange les éléments de valeur égale à celle du pivot avec l'élément d'indice a et on stoppe sur un élément strictement plus grand. D'une façon similaire, la deuxième boucle interne décrémente c : on passe sur les éléments de valeur strictement supérieure à la valeur du pivot, on échange les éléments de valeur égale à celle du pivot en mettant à jour l'indice d, et on s'arrête sur un élément de valeur strictement inférieure. Ensuite on revient dans la boucle principale de partitionnement qui échange les éléments pointés par b et c. On itère ce processus

```
void swap(char **tab, int i, int j ) {
   char *t;
   t = tab[i];
   tab[i] = tab[j];
   tab[j] = t;
}

void vecswap(int i, int j, int n, char **tab) {
   while (n-- >0) {
      swap(tab, i, j);
      i++;
      j++;
   }
}
```

Code 5.10: Fonction d'échange de deux chaînes *via* leurs indices et généralisation déplaçant un ensemble de chaînes de caractères.

jusqu'à ce que les pointeurs b et c se croisent. Ensuite les éléments de valeur égale au pivot sont déplacés au centre du tableau sans comparaisons supplémentaires. Cette procédure de partitionnement s'effectue à l'aide de $n-1$ comparaisons. Plus important, le principe de partitionnement va s'appliquer sur des tableaux qui ne contiennent plus la valeur du pivot. Il y a là un gros avantage par rapport à la procédure de partitionnement classique s'il y a beaucoup de clés de même valeur. On utilise une fonction qui (voir le code 5.10) échange deux chaînes (d'indices `i` et `j`) dans un tableau de chaînes (en manipulant les pointeurs), et l'on étend cette fonction pour échanger une portion complète du tableau de chaînes. Le programme de tri qui résulte des principes exposés ci-dessus correspond au code 5.11.

```
void sort(char **tab, int n, int depth) {
   int a, b, c, d, r, v;
   if (n <= 1)
      return;
   v = tab[0][depth];
   a = b = 1;
   c = d = n-1;
   for (;;) {
      while (b <= c && (r = tab[b][depth] - v) <=0) {
         if (r == 0) {
            swap(tab, a, b);
            a++;
         }
         b++;
      }
      while (b <= c && (r = tab[c][depth] - v) >=0) {
         if (r == 0) {
            swap(tab, c, d);
            d--;
         }
         c--;
      }
      if (b > c) break;
      swap(tab, b, c);
      b++;
      c--;

      r = min(a, b-a);
      vecswap(0, b-r, r, tab);
      r= min(d-c, n-d-1);
      vecswap(b, n-r, r, tab);
      r=b-a;
      sort(tab,r,depth);
      if (tab[r][depth] != 0)
         sort(tab+r, a + n-d-1, depth+1);
      r=d-c;
      sort(tab+n-r, r, depth);
   }
}
```

Code 5.11: Fonction du tri «quick-radix»

Chapitre 6

Structures de recherche aux nœuds

Sommaire

On présente les outils nécessaires à l'étude des structures de recherche utilisées dans cette thèse, c.-à-d. les séries génératrices ordinaires et exponentielles, avec les liens très forts qui les unissent aux langages formels et à différents modèles aléatoires. Enfin, les structures de liste triée et d'arbre binaire de recherche sont étudiées plus en détail en vue de l'analyse des tries hybrides.

On a montré dans le chapitre précédent que les paramètres additifs de trie admettent la décomposition suivante pour les espérances

$$\mathrm{E}[\gamma; \mathcal{P}_z, \mathsf{S}, F] = \sum_{\boldsymbol{w} \in \mathcal{M}^*} \mathrm{E}[\delta; \mathcal{P}_{z u_{\boldsymbol{w}}}, \{p_{\boldsymbol{w},m}\}], \tag{6.1}$$

où $\boldsymbol{w}$ décrit $\mathcal{M}^*$ l'ensemble des mots (finis), et P_z désigne le modèle de Poisson de paramètre z.

Si le modèle de Poisson apparaît deux fois dans la formule (6.1), il n'a pas la même signification des deux cotés de l'égalité. Le modèle de Poisson du membre de gauche est relatif au nombre de mots infinis émis par une source dynamique probabilisée (la taille est donc le cardinal de l'ensemble des mots). Le modèle de Poisson du membre de droite est relatif à la taille de mots finis (c.-à-d. la longueur des mots), et les mots sont émis par une source sans mémoire.

Nous nous intéressons maintenant au deuxième modèle, celui des mots finis produits par une source sans mémoire. Du fait des structures envisagées (listes et arbres binaires de recherche), les

mots sont décomposables grâce à des opérations sur des langages formels. L'utilisation de séries génératrices s'appuyant sur ces décompositions permet d'obtenir les informations recherchées sur les paramètres.

6.1 Langages formels, séries génératrices multivariées

Cette section commence par une présentation courte de quatre opérations de base sur les langages. On définit ensuite les séries génératrices ordinaires et exponentielles multivariées. Enfin on explique de quelle façon les opérations sur des langages se traduisent directement sur les séries génératrices.

6.1.1 Langages

Soit un alphabet fixé $\mathcal{M} = \{a_1, a_2, \ldots, a_r\}$ un alphabet fini. L'ensemble des mots finis construit sur $\mathcal{M}$ est $\mathcal{M}^*$. Un langage L est un sous-ensemble de $\mathcal{M}^*$. Nous utiliserons dans le cadre de notre analyse les opérations suivantes sur les langages.

– *Union.* L'union de deux langages L_1 et L_2 (noté $L_1 + L_2$) est définie par

$$L_1 + L_2 = \{\boldsymbol{w} \mid \boldsymbol{w} \in L_1 \text{ ou } \boldsymbol{w} \in L_2\}.$$

– *Concaténation.* Le concaténé de deux langages L_1 et L_2 est le langage obtenu en concaténant chacun des mots $\boldsymbol{w_1}$ de L_1 avec chacun des mots $\boldsymbol{w_2}$ de L_2

$$L_1 \cdot L_2 = \{\boldsymbol{w}_1\boldsymbol{w}_2 \,|\, \boldsymbol{w}_1 \in L_1,\ \boldsymbol{w}_2 \in L_2\}.$$

– *«Étoile».* Le langage L^* est le langage obtenu en formant toutes les suites finies d'éléments de L

$$L^* = \{\epsilon\} + L + (L \cdot L) + (L \cdot L \cdot L) + \cdots, \text{ où } \epsilon \text{ désigne le mot vide.}$$

– *Shuffle.* Le «shuffle» (voir [36, 70]) de deux mots $\boldsymbol{w}_1$ et $\boldsymbol{w}_2$, noté $(\boldsymbol{w}_1 \sqcup\!\sqcup \boldsymbol{w}_2)$ est *l'ensemble* de tous les mots obtenus en mélangeant[1] de toutes les façons possibles les lettres de $\boldsymbol{w}_1$ et $\boldsymbol{w}_2$, tout en préservant l'ordre relatif à l'intérieur de $\boldsymbol{w}_1$ et $\boldsymbol{w}_2$. Cette opération est définie récursivement par

$$(a\boldsymbol{v}_1 \sqcup\!\sqcup b\boldsymbol{v}_2) = a(\boldsymbol{v}_1 \sqcup\!\sqcup b\boldsymbol{v}_2) \cup b(a\boldsymbol{v}_1 \sqcup\!\sqcup \boldsymbol{v}_2),$$

avec $(\boldsymbol{v} \sqcup\!\sqcup \epsilon) = (\epsilon \sqcup\!\sqcup \boldsymbol{v}) = \{\boldsymbol{v}\}$. L'opération de «shuffle» n'est envisagée ici que sur deux langages d'alphabets disjoints. (Dans ce cas, on peut reconstruire de manière unique les mots $\boldsymbol{w_1}$ et $\boldsymbol{w}_2$ à partir d'un mot de $\boldsymbol{w}_1 \sqcup\!\sqcup \boldsymbol{w}_2$.) Par exemple, on a $(ab \sqcup\!\sqcup cd) = \{abcd, acbd, acdb, cabd, cadb, cdab\}$.
Appliquée à deux langages L_1 et L_2 d'alphabets distincts, l'opération de «shuffle» permet d'obtenir le langage suivant

$$L_1 \sqcup\!\sqcup L_2 = \bigcup_{\substack{\boldsymbol{w}_1 \in L_1 \\ \boldsymbol{w}_2 \in L_2}} (\boldsymbol{w}_1 \sqcup\!\sqcup \boldsymbol{w}_2).$$

Dans la suite, on désignera par $\mathsf{alph}(L)$ et $\mathsf{alph}(\boldsymbol{w})$ l'ensemble des lettres distinctes apparaissant respectivement dans un langage L ou un mot $\boldsymbol{w}$.

[1] La meilleure image est celle de deux jeux de cartes que l'on mélange, les mots $\boldsymbol{w}_1$ et $\boldsymbol{w}_2$ étant chacun un jeu de cartes.

6.1.2 Séries génératrices

Nous avons déjà présenté les séries génératrices dans le cadre de l'analyse «classique» des tries. Il s'agit ici d'étendre la définition au cas multivarié. Les séries génératrices du chapitre 3 étaient essentiellement bivariées, la variable z «marquant» la taille et la variable u «marquant» un autre paramètre. Les séries génératrices sont ici des séries génératrices relatives à des langages. Pour chaque mot du langage, en plus de la taille du mot et du paramètre qu'on veut étudier, on va également marquer le nombre d'occurrences de chaque lettre.

Série génératrice ordinaire et exponentielle

La série génératrice ordinaire (SGO en abrégé) d'un paramètre δ (correspondant à la variable formelle u) sur un langage L est

$$\ell(z, u, x_1, \ldots, x_r) = \sum_{\boldsymbol{w}\in L} u^{\delta(\boldsymbol{w})} z^{|\boldsymbol{w}|} x_1^{|\boldsymbol{w}|_1} \cdots x_r^{|\boldsymbol{w}|_r}, \tag{6.2}$$

où $|\boldsymbol{w}|$ est la longueur du mot $\boldsymbol{w}$ et $|\boldsymbol{w}|_i$ est le nombre d'occurrences de la lettre a_i dans $\boldsymbol{w}$. La série génératrice énumérative d'un langage correspond au cas particulier où le paramètre δ est la fonction identiquement nulle.
De manière similaire, la série génératrice exponentielle (SGE en abrégé) d'un paramètre δ pour un langage L est

$$\widehat{\ell}(z, u, x_1, \ldots, x_r) = \sum_{\boldsymbol{w}\in L} u^{\delta(\boldsymbol{w})} \frac{z^{|\boldsymbol{w}|}}{|\boldsymbol{w}|!} x_1^{|\boldsymbol{w}|_1} \cdots x_r^{|\boldsymbol{w}|_r}. \tag{6.3}$$

C'est souvent la nature du problème qui incite à utiliser la version exponentielle plutôt qu'ordinaire.
Les séries cumulées associées aux séries génératrices ordinaires et exponentielles sont obtenues en dérivant par rapport à u et en posant $u = 1$,

$$C_\delta(z, x_1, \ldots, x_r) = \frac{\partial}{\partial u}\, \ell(z, u, x_1, \ldots, x_r)|_{u=1} = \sum_{\boldsymbol{w}\in L} \delta(\boldsymbol{w}) z^{|\boldsymbol{w}|} x_1^{|\boldsymbol{w}|_1} \cdots x_r^{|\boldsymbol{w}|_r}$$
$$\widehat{C}_\delta(z, x_1, \ldots, x_r) = \frac{\partial}{\partial u}\, \widehat{\ell}(z, u, x_1, \ldots, x_r)\Big|_{u=1} = \sum_{\boldsymbol{w}\in L} \delta(\boldsymbol{w}) \frac{z^{|\boldsymbol{w}|}}{|\boldsymbol{w}|!} x_1^{|\boldsymbol{w}|_1} \cdots x_r^{|\boldsymbol{w}|_r}.$$

La transformée de Laplace combinatoire permet de relier de manière naturelle les séries génératrices exponentielles et ordinaires. Elle est définie par additivité à partir de la relation

$$\mathcal{L} : z^n \mapsto \frac{z^n}{n!}.$$

La relation classique suivante est typique de la façon dont on passe du «monde ordinaire» au «monde exponentiel»

$$\mathcal{L}\left[\frac{1}{1-az}\right] = \mathrm{e}^{az}.$$

Dans la suite, on désignera par $\mathcal{L}F(z)$ ou $\widehat{F}(z)$ la transformée de Laplace combinatoire d'une série génératrice ordinaire $F(z)$.

6.1.3 Langages et séries génératrices multivariées

THÉORÈME 6.1. *Soit un alphabet $\mathcal{M} = \{a_1, a_2, \ldots, a_r\}$. On désigne par $x_1, \ldots, x_r$ les variables formelles marquant les occurrences de chaque lettre, par δ un paramètre additif sur $\mathcal{M}^*$. On considère deux langages L_1 et L_2 d'alphabets $\mathsf{alph}(L_1)$ et $alph(L_2)$ inclus dans $\mathcal{M}$.*

– *Union. Si $L_1 \cap L_2 = \varnothing$, le langage $L = L_1 + L_2$, obtenu en faisant l'union, admet pour séries génératrices*

$$\begin{cases} \ell(z,u,x_1,\ldots,x_r) = \ell_1(z,u,x_1,\ldots,x_r) + \ell_2(z,u,x_1,\ldots,x_r) & (\text{SGO}) \\ \widehat{\ell}(z,u,x_1,\ldots,x_r) = \widehat{\ell}_1(z,u,x_1,\ldots,x_r) + \widehat{\ell}_2(z,u,x_1,\ldots,x_r) & (\text{SGE}) \end{cases}.$$

– *Concaténation. Le langage $L = L_1 \cdot L_2$ obtenu par concaténation a pour série génératrice ordinaire*

$$\ell(z,u,x_1,\ldots,x_r) = \ell_1(z,u,x_1,\ldots,x_r) \cdot \ell_2(z,u,x_1,\ldots,x_r) \quad (\text{SGO}).$$

– *Opération «étoile». Si $L_1 \cap (L_1 \cdot L_1) = \varnothing$, le langage $L = L_1^*$ obtenu par l'opération «étoile» d'un langage L_1 a pour série génératrice ordinaire*

$$\ell(z,u,x_1,\ldots,x_r) = (1 - \ell_1(z,u,x_1,\ldots,x_r))^{-1} \quad (\text{SGO}).$$

Cas particulier important : *Si le langage $L_1 \subset \mathcal{M}$, i.e. n'est composé que de mots de longueur 1, on obtient une expression de la série génératrice exponentielle pour le langage $L = L_1^*$*

$$\widehat{\ell}(z,u,x_1,\ldots,x_r) = \exp(\widehat{\ell}_1(z,u,x_1,\ldots,x_r)) \quad (\text{SGE}).$$

– *Shuffle. Supposons que $\mathsf{alph}(L_1) \cap \mathsf{alph}(L_2) = \varnothing$. Le langage $L = L_1 \sqcup\!\sqcup L_2$ obtenu par l'opération shuffle sur les langages L_1 et L_2 a pour série génératrice exponentielle*

$$\widehat{\ell}(z,u,x_1,\ldots,x_r) = \widehat{\ell}_1(z,u,x_1,\ldots,x_r) \cdot \widehat{\ell}_2(z,u,x_1,\ldots,x_r) \quad (\text{SGE}).$$

Preuve. La preuve est immédiate pour l'union et s'obtient par linéarité.

Pour la concaténation, considérons deux langages L_1 et L_2 de séries génératrices ordinaires $\ell_1(z,u,x_1,\ldots,x_r)$ et $\ell_2(z,u,x_1,\ldots,x_r)$. La série génératrice $\ell(z,u,x_1,\ldots,x_r)$ est définie par

$$\begin{aligned} \ell(z,u,x_1,\ldots,x_r) &= \sum_{\boldsymbol{w}_1 \in L_1} \sum_{\boldsymbol{w}_2 \in L_2} u^{\delta(\boldsymbol{w}_1\boldsymbol{w}_2)} z^{|\boldsymbol{w}_1|+|\boldsymbol{w}_2|} x_1^{|\boldsymbol{w}_1|_1+|\boldsymbol{w}_2|_1} \cdots x_r^{|\boldsymbol{w}_1|_r+|\boldsymbol{w}_2|_r} \\ &= \left(\sum_{\boldsymbol{w}_1 \in L_1} u^{\delta(\boldsymbol{w}_1)} z^{|\boldsymbol{w}_1|} x_1^{|\boldsymbol{w}_1|} \cdots x_r^{|\boldsymbol{w}_1|_r} \right) \left(\sum_{\boldsymbol{w}_2 \in L_2} u^{\delta(\boldsymbol{w}_2)} z^{|\boldsymbol{w}_2|} x_1^{|\boldsymbol{w}_2|} \cdots x_r^{|\boldsymbol{w}_2|_r} \right) \\ &= \ell_1(z,u,x_1,\ldots,x_r)\ \ell_2(z,u,x_1,\ldots,x_r). \end{aligned}$$

La deuxième égalité utilise le fait que δ est un paramètre additif ($\delta(\boldsymbol{w}_1\boldsymbol{w}_2) = \delta(\boldsymbol{w}_1) + \delta(\boldsymbol{w}_2)$).

La série génératrice ordinaire pour le langage «étoile» est calculée grâce aux formules pour l'union et la concaténation.

Dans le cas où L_1 ne contient que des symboles ($L_1 \subset \mathcal{M}$), Le langage L_1^k admet comme série génératrice exponentielle

$$\begin{aligned} \sum_{\boldsymbol{w} \in L_1^k} u^{\delta(\boldsymbol{w})} \frac{z^{|\boldsymbol{w}|}}{|\boldsymbol{w}|!} x_1^{|\boldsymbol{w}|} \cdots x_r^{|\boldsymbol{w}|_r} &= \sum_{m_1,\ldots,m_k \in L_1} u^{\delta(m_1 m_2 \ldots m_k)} \frac{z^k}{k!} x_1^{|m_1 m_2 \ldots m_k|} \cdots x_r^{|m_1 m_2 \ldots m_k|_r} \\ &= \frac{z^k}{k!} \left(\sum_{m \in L_1} u^{\delta(m)} x_1^{|m|_1} \cdots x_r^{|m|_r} \right)^k = \frac{z^k}{k!} \widehat{\ell}_1(1,u,x_1,\ldots,x_r)^k \\ &= \frac{\widehat{\ell}_1(z,u,x_1,\ldots,x_r)^k}{k!}, \end{aligned}$$

et la formule s'ensuit pour le langage $L = \sum_{k\geq 0} L_1^k$

$$\widehat{\ell} = \exp[\widehat{\ell}_1(z, u, x_1, \ldots, x_r)].$$

Enfin, pour le shuffle, considérons deux mots $\boldsymbol{w}_1$ et $\boldsymbol{w}_2$ d'alphabets disjoints. La série génératrice exponentielle du langage $L = \boldsymbol{w}_1 \sqcup\!\sqcup \boldsymbol{w}_2$ fait intervenir un coefficient du binôme correspondant aux places où on insère $\boldsymbol{w}_1$ dans $\boldsymbol{w}_2$

$$\begin{aligned}\widehat{\ell}(z, u, x_1, \ldots, x_r) &= \frac{(|\boldsymbol{w}_1| + |\boldsymbol{w}_2|)!}{|\boldsymbol{w}_1|!|\boldsymbol{w}_2|!} \frac{z^{|\boldsymbol{w}_1|+|\boldsymbol{w}_2|}}{(|\boldsymbol{w}_1| + |\boldsymbol{w}_2|)!} x_1^{|\boldsymbol{w}_1|_1+|\boldsymbol{w}_2|_1} \cdots x_r^{|\boldsymbol{w}_1|_r+|\boldsymbol{w}_2|_r} \\ &= \left(\frac{z^{|\boldsymbol{w}_1|}}{|\boldsymbol{w}_1|!} x_1^{|\boldsymbol{w}_1|_1} \cdots x_r^{|\boldsymbol{w}_1|_r}\right) \left(\frac{z^{|\boldsymbol{w}_2|}}{|\boldsymbol{w}_2|!} x_1^{|\boldsymbol{w}_2|_1} \cdots x_r^{|\boldsymbol{w}_2|_r}\right).\end{aligned}$$

Par linéarité de la formule de l'union de langages d'intersection vide, on obtient l'expression de la série génératrice exponentielle pour le shuffle de deux langages

$$\begin{aligned}\widehat{\ell}(z, u, x_1, \ldots, x_r) &= \sum_{\boldsymbol{w}_1 \in L_1} \sum_{\boldsymbol{w}_2 \in L_2} \left(\frac{z^{|\boldsymbol{w}_1|}}{|\boldsymbol{w}_1|!} x_1^{|\boldsymbol{w}_1|_1} \cdots x_r^{|\boldsymbol{w}_1|_r}\right) \left(\frac{z^{|\boldsymbol{w}_2|}}{|\boldsymbol{w}_2|!} x_1^{|\boldsymbol{w}_2|_1} \cdots x_r^{|\boldsymbol{w}_2|_r}\right) \\ &= \widehat{\ell}_1(z, u, x_1, \ldots, x_r)\ \widehat{\ell}_2(z, u, x_1, \ldots, x_r),\end{aligned}$$

ce qui termine la preuve. □

Remarque – Il y a une symétrie frappante entre l'opération de concaténation et l'opération de «shuffle» du point de vue des séries génératrices. Le «shuffle» est une opération naturelle pour les séries génératrices exponentielles car cette opération se traduit par le produit des deux SGE.

La concaténation est une opération naturelle pour les séries génératrices ordinaires car cette opération se traduit par le produit des deux SGO.

Cependant, si la nature du problème se prête davantage à l'utilisation de *séries génératrices ordinaires* ou de *séries génératrices exponentielles*, nous pouvons toujours définir l'opération de shuffle pour deux SGO et de concaténation pour deux SGE. Il suffit d'utiliser la transformée de Laplace combinatoire pour naviguer entre les «mondes» exponentiels et ordinaires. Ainsi, l'opération de shuffle pour des séries génératrices ordinaires peut s'écrire

$$\begin{aligned}&A(z, u, x_1, \ldots, x_p) \sqcup\!\sqcup B(z, u, y_1, \ldots, y_q) \\ &= \mathcal{L}^{-1}\left[\mathcal{L}A(z, u, x_1, \ldots, x_p) \cdot \mathcal{L}B(z, u, y_1, \ldots, y_q)\right] \\ &= \mathcal{L}^{-1}\left[\widehat{A}(z, u, x_1, \ldots, x_p) \cdot \widehat{B}(z, u, y_1, \ldots, y_q)\right].\end{aligned}$$

Nous utilisons deux ensembles distincts de variables $\{x_1, \ldots, x_p\}$ et $\{y_1, \ldots, y_q\}$ pour insister sur le fait que les deux séries génératrices font référence à des langages d'alphabets distincts, ce qui rend l'opération de shuffle non ambiguë. L'opération de shuffle appliquée à deux séries génératrices ordinaires rationnelles s'écrit

$$\frac{1}{1-az} \sqcup\!\sqcup \frac{1}{1-bz} = \frac{1}{1-(a+b)z}.$$

L'analyse nécessitera également de dériver un produit shuffle sur tout type de série, afin d'obtenir

la série cumulée. Par exemple, le produit shuffle de deux séries ordinaires s'obtient grâce à la règle

$$\begin{aligned}&\frac{\partial}{\partial u}\left(A(z,u,x_1,\dots,x_p)\,{\sqcup\!\sqcup}\,B(z,u,y_1,\dots,y_q)\right)\\ =&\left(\frac{\partial}{\partial u}A(z,u,x_1,\dots,x_p)\right)\,{\sqcup\!\sqcup}\,B(z,u,y_1,\dots,y_q)+\\ &A(z,u,x_1,\dots,x_p)\,{\sqcup\!\sqcup}\,\left(\frac{\partial}{\partial u}B(z,u,y_1,\dots,y_q)\right).\end{aligned}$$

6.1.4 Séries génératrices et modèles aléatoires

Nous examinons ici plusieurs modèles aléatoires en relation avec les séries génératrices multivariées. Le premier modèle est le modèle du multi-ensemble. C'est le modèle le plus proche de la série génératrice puisqu'il s'agit d'extraire un coefficient de la série génératrice.

DÉFINITION 6.2 (MODÈLE DU MULTI-ENSEMBLE). *Dans le modèle du multi-ensemble $\mathcal{E}_{n_1,\dots,n_r}$, on considère comme équiprobables tous les mots de taille $n = n_1 + \cdots + n_r$ qui contiennent n_i occurrences du symbole a_i.*

Exemple. Le multi-ensemble $\{2 \cdot a, 1 \cdot b\}$ est l'ensemble $\{aab, aba, baa\}$. Chacun des trois mots est équiprobable dans le modèle aléatoire du multi-ensemble. Ce modèle va réapparaître sous une autre forme dans le chapitre 7.

Dans un tel modèle, les mots sont donc en nombre $\binom{n}{n_1, n_2, \dots, n_r}$.

Toute la mécanique des séries génératrices du chapitre 3 peut être appliquée. Ainsi, l'espérance d'un paramètre δ sur un langage L (de série génératrice ordinaire associée ℓ par exemple) dans le modèle aléatoire $\mathcal{E}_{n_1,\dots,n_r}$ s'obtient grâce à la formule

$$\frac{[z^n x_1^{n_1}\cdots x_r^{n_r}]\frac{\partial}{\partial u}\,\ell(z,u,x_1,\dots,x_r)|_{u=1}}{[z^n x_1^{n_1}\cdots x_r^{n_r}]\ell(z,1,x_1,\dots,x_r)}.$$

Nous allons en fait plutôt nous intéresser à des modèles «plus» probabilistes où nous ne connaîtrons pas le nombre d'occurrences de chaque lettre mais plutôt la probabilité d'apparition de chaque lettre.

Puisque les modèles considérés concernent des mots, on distingue deux aspects différents.

– Le premier concerne la taille du mot. La taille du mot peut être une variable aléatoire N d'une certaine loi. La loi la plus simple consiste à fixer la taille $N = n$. C'est le modèle de Bernoulli $\mathcal{B}_n$. Une autre possibilité consiste à prendre pour N une loi de Poisson de paramètre z, ce qui désigne le modèle de Poisson $\mathcal{P}_z$. (Ces deux modèles ont déjà été succinctement introduits dans le chapitre 5 et une discussion plus détaillée sur les rapports entre les deux modèles d'un point de vue analytique se trouve dans la partie 6.5.1.)
– Une fois la taille précisée, il reste à définir de quelle façon sont produits les mots. Les mots sont ici finis et émis par une source sans mémoire avec des probabilités $\{p_i\}_{i\in\mathcal{M}}$ (voir chapitre 4).

DÉFINITION 6.3 (MODÈLE DE BERNOULLI POUR LES MOTS). *Dans ce modèle, la taille des mots n est fixée. Les lettres d'un mot sont tirées aléatoirement dans l'alphabet avec des probabilités p_i (comme par une source sans mémoire).*

La substitution $x_j \mapsto p_j$ dans la série génératrice ordinaire $\ell(z, u, x_1, \ldots, x_r)$ associée à $\mathcal{M}^*$ permet d'obtenir la série génératrice (bivariée puisque les p_j ne sont plus des variables) $\ell(z, u)$ du paramètre δ dans le modèle de Bernoulli

$$\ell(z,u) = \sum_{\boldsymbol{w}\in\mathcal{M}^*} u^{\delta(\boldsymbol{w})} z^{|\boldsymbol{w}|} p_1^{|\boldsymbol{w}|_1} \cdots p_r^{|\boldsymbol{w}|_r} = \sum_{\boldsymbol{w}\in\mathcal{M}^*} u^{\delta(\boldsymbol{w})} z^{|\boldsymbol{w}|} \Pr[\boldsymbol{w}],$$

où $\Pr[\boldsymbol{w}]$ est la probabilité d'apparition du mot $\boldsymbol{w}$ (parmi les mots de même longueur).

Remarque : On peut également considérer l'alphabet infini. En effet, dans la formule

$$\sum_{\boldsymbol{w}\in\mathcal{M}^*} u^{\delta(\boldsymbol{w})} z^{|\boldsymbol{w}|} \prod_{i\in\mathcal{M}} p_i^{|\boldsymbol{w}|_i}$$

le produit infini a un sens puisqu'il n'y a qu'un nombre fini de $|\boldsymbol{w}|_i$ non nuls.

Ainsi, on obtient la proposition faisant intervenir la série génératrice cumulée du paramètre δ

$$C_\delta(z) = \frac{\partial}{\partial u}\, \ell(z, u, p_1, \ldots, p_r)|_{u=1}\,.$$

PROPOSITION 6.4 (VALEUR MOYENNE DANS LE MODÈLE DE BERNOULLI). *Dans le modèle de Bernoulli $\mathcal{B}_n$ associé à une source sans mémoire de probabilités $\{p_j\}$, la valeur moyenne du paramètre δ pour un mot de longueur n est*

$$\mathrm{E}[\delta; \mathcal{B}_n, \{p_j\}] = [z^n]\, \frac{\partial}{\partial u}\, \ell(z, u, p_1, \ldots, p_r)|_{u=1} = [z^n] C_\delta(z),$$

où ℓ est la série génératrice ordinaire associée au paramètre δ et au langage $\mathcal{M}^*$, et la notation $[z^n]\, f(z)$ signifie le coefficient en z^n de la série entière $f(z)$.

Les séries exponentielles sont adaptées au deuxième modèle dit de Poisson. En particulier, l'espérance dans le modèle de Poisson s'exprime de manière directe.

Rappelons que, par définition, une variable aléatoire de Poisson N de paramètre z vérifie :

$$Pr[N = k] = \mathrm{e}^{-z}\, \frac{z^k}{k!}. \tag{6.4}$$

On a donc $\mathrm{E}[N] = z$ et $\mathrm{Var}[N] = z$.

DÉFINITION 6.5 (MODÈLE DE POISSON POUR LES MOTS). *Dans le modèle de Poisson $\mathcal{P}_z$ de paramètre z, la longueur d'un mot aléatoire est N, elle-même une variable de Poisson de paramètre z. Les lettres d'un mot sont tirées aléatoirement dans l'alphabet avec des probabilités p_i (comme par une source sans mémoire).*

Si l'on pose $z = n$, ce modèle est «proche» en moyenne du modèle de Bernoulli à taille fixée n (même s'il est moins naturel), tout en permettant un traitement théorique plus aisé. En fait on passe d'un modèle à l'autre en appliquant poissonisation et dépoissonisation [57] (voir à ce sujet également la partie 6.5.1).

Soit δ un paramètre entier sur le langage $L = \mathcal{M}^*$. Posons $\delta_k = \mathrm{E}[\delta, \mathcal{B}_k, \{p_i\}]$ de telle sorte que $\widehat{C}_\delta(z) = \sum_{k\geq 0} \frac{\delta_k}{k!} z^k$ est la série génératrice exponentielle cumulée du paramètre δ. L'espérance dans le modèle de Poisson vérifie en vertu de (6.4)

$$\mathrm{E}[\delta, \mathcal{P}_z, \{p_j\}] = \sum_{k\geq 0} \frac{\delta_k}{k!} z^k = \widehat{C}_\delta(z).$$

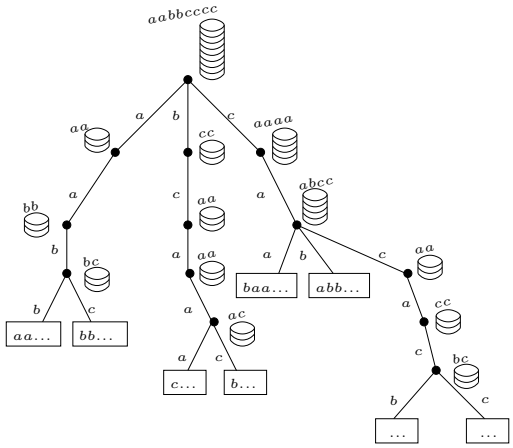


FIG. 6.1 – On donne au trie une épaisseur en empilant des jetons sur chaque nœud interne. L'ensemble de mots est le même que celui de la figure 2.2 ($aabbaa...$, $aabcbb...$, $bcaaac...$, $bcaacb...$, $caabaa...$, $cababb...$, $caccbb...$, $caccbc...$). Par exemple, on empile sur la racine la première lettre de chaque mot $aabbcccc$.

On a donc la proposition

PROPOSITION 6.6 (ESPÉRANCE DANS LE MODÈLE DE POISSON). *Dans le modèle de Poisson $\mathcal{P}_z$ de paramètre z associé à une source sans mémoire de probabilités $\{p_j\}$, la valeur moyenne du paramètre δ pour un mot de longueur n est*

$$\mathrm{E}[\delta, \mathcal{P}_z, \{p_j\}] = e^{-z} \frac{\partial}{\partial u} \widehat{\ell}(z, u, p_1, \ldots, p_r))\Big|_{u=1} = e^{-z} \widehat{C}_\delta(z), \tag{6.5}$$

où $\widehat{\ell}$ est la série génératrice exponentielle associée au paramètre δ et au langage $\mathcal{M}^*$.

6.2 Étude des paramètres du trie abstrait

Nous considérons maintenant les trois paramètres additifs, de type γ, dans l'équation (5.3) reliés respectivement au nombre de nœuds internes (appelé également taille du trie), la longueur de cheminement du trie-tableau. (La longueur de cheminement dans le trie-ABR, plus difficile à obtenir, fera l'objet de la partie suivante. Nous établirons la longueur de cheminement du trie-liste comme un prélude à celle du trie-ABR.)

Il faut donc préciser ce qu'est le coût δ associé à la traversée d'une structure et, utilisant les outils introduits dans la section précédente, en calculer les espérances.

Toute cette partie demande d'adopter un point de vue un peu particulier sur les tries mais qui éclaire notre approche. Il consiste à visualiser les tries en donnant aux nœuds des épaisseurs. Imaginons des jetons étiquetés par les lettres de l'alphabet (lettres de *Scrabble* par exemple). On empile sur chaque nœud étiqueté $\boldsymbol{w}$ les jetons correspondant à la tranche $\sigma(T_{[\boldsymbol{w}]}(X))$ (voir figure 6.1). Par exemple la racine est une pile de jetons de hauteur $|X|$, et que chacun des nœuds externes n'a pas de jetons. Les paramètres de taille et de longueur de cheminement ont une interprétation directe dans ce cadre. La longueur de cheminement est le nombre de jetons posés sur le trie ; la taille du trie est le nombre de piles.

Tout d'abord, le coût δ_S associé à la taille est égal à 1 si le nœud interne indexé par le mot $\boldsymbol{w}$ existe ou de façon équivalente si la tranche $\sigma T_{[\boldsymbol{w}]}(X)$ contient au moins deux symboles (et donc aussi si la pile est de hauteur supérieure ou égale à deux). Le coût δ_A pour la longueur de cheminement pour un trie-tableau est simplement le nombre de fois où l'on accède à un nœud (le nombre de symboles de $\sigma T_{[\boldsymbol{w}]}(X)$) si le nœud est interne[2]. C'est donc la hauteur de la pile de jetons sur ce nœud interne. Soit $\boldsymbol{s}$ une tranche (ou une pile), i.e. un mot rassemblant toutes les premières lettres des chaînes stockées en un nœud interne. On a les formules

$$\delta_S(\boldsymbol{s}) = \begin{cases} 1 & \text{si } |\boldsymbol{s}| \geq 2 \\ 0 & \text{sinon} \end{cases} \quad \text{et} \quad \delta_A(\boldsymbol{s}) = \begin{cases} |\boldsymbol{s}| & \text{si } |\boldsymbol{s}| \geq 2 \\ 0 & \text{sinon.} \end{cases}$$

Les résultats suivants sont connus et ont d'ailleurs déjà été démontrés dans le modèle de Bernoulli dans le chapitre 5. Dans le but de donner un exemple d'analyse simple utilisant les méthodes symboliques des séries génératrices, nous établissons la proposition suivante pour les valeurs moyennes des paramètres δ_A et δ_S.

PROPOSITION 6.7 (PARAMÈTRES DE STRUCTURE DU TRIE ABSTRAIT (POISSON)). *Soit un ensemble de probabilités $\{p_i\}_{i\in\mathcal{M}}$ associé aux lettres de l'alphabet $\mathcal{M}$ et $\mathcal{P}_z$ le modèle de Poisson de paramètre z pour la taille des mots. Alors, dans le modèle $(\mathcal{P}_z, \{p_i\})$, les espérances des paramètres de «péage» pour la taille d'un trie et la longueur de cheminement externe d'un trie-tableau*[3] *sont respectivement*

$$\mathrm{E}[\delta_S; \mathcal{P}_z, \{p_i\}] = 1 - (1+z)\,\mathrm{e}^{-z}, \qquad \mathrm{E}[\delta_A; \mathcal{P}_z, \{p_i\}] = z\,(1-\mathrm{e}^{-z}).$$

Preuve. Les espérances des deux premiers paramètres, δ_S et δ_A, dérivent directement de propriétés du modèle de Poisson. Nous effectuons cependant l'analyse dans le cadre des séries génératrices qui permet de mettre en lumière quelques principes de base. La décomposition formelle du langage $\mathcal{M}^*$ selon la taille des mots

$$\mathcal{M}^* = (\epsilon + \mathcal{M}) + \sum_{i\geq 2} \mathcal{M}^k,$$

se traduit en SGE (séries génératrices exponentielles) grâce au théorème 6.1 et donne

$$1 + z(x_1 + \cdots + x_r) + u(\mathrm{e}^{z(x_1+\cdots+x_r)} - 1 - z(x_1 + \cdots + x_r)),$$
$$1 + z(x_1 + \cdots + x_r) + (\mathrm{e}^{zu(x_1+\cdots+x_r)} - 1 - zu(x_1 + \cdots + x_r)),$$

pour les SGE relatives à δ_S et δ_A. Une application de (6.5) donne alors la formule pour $\mathrm{E}[\delta_A]$ et $\mathrm{E}[\delta_S]$. □

Remarques.
– On n'utilise pas le fait que les jetons soient étiquetés par des lettres. Seul le nombre de jetons sur un nœud interne importe.
– On ne donne pas le corollaire pour le modèle de Bernoulli car il est évident.

6.3 Arbres binaires de recherche

Les arbres binaires sont une structure de données extrêmement classique et utilisée. Il est classique de «lire» le déroulement de l'algorithme Quicksort sur un arbre binaire de recherche. Les modèles

[2] Ce coût correspond à l'accès direct.
[3] Il s'agit donc aussi de la longueur de cheminement d'un trie classique.

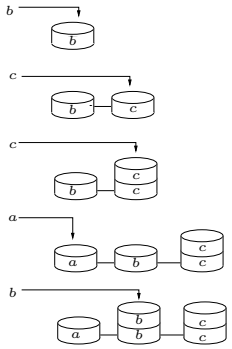


FIG. 6.2 – Mécanisme d'empilement des jetons dans le cas d'une liste ordonnée, on insère dans l'ordre et successivement les lettres $bccab$.

d'analyse sont longtemps restés centrés autour du modèle des permutations aléatoires (qui est un cas particulier du modèle du multi-ensemble $\mathcal{E}_{n_1,\ldots,n_r}$ avec $n_1 = \cdots = n_r = 1$) qui est souvent le premier modèle envisagé pour l'analyse en moyenne des tris. Les techniques de séries génératrices s'appliquent et l'on peut démontrer par exemple que pour une permutation de $\mathcal{S}_r$ la longueur de cheminement interne (pour atteindre chacune des clés à partir de la racine)

$$2(n+1)H_n - 4n,$$

où H_n désigne le n^{e} nombre harmonique.
Les résultats concernant la profondeur du plus petit élément et du plus grand élément nous intéressent également et peuvent être très facilement obtenus à partir de la proposition 6.8.

Helmut Prodinger [84] s'est intéressé à un modèle où les clés apparaissent avec une distribution géométrique.

Le modèle du multi-ensemble a également été étudié par plusieurs auteurs [14, 89] mais toujours avec une approche calculatoire. Les techniques mises en œuvre dans cette partie sont essentiellement symboliques et font intervenir les séries génératrices multivariées.

Les paramètres $\delta_L(\boldsymbol{s})$ et $\delta_B(\boldsymbol{s})$ relatifs à la longueur de cheminement pour les tries-liste et les tries-ABR sont exactement les coûts de traversée dans la structure nœud construite sur la tranche $\boldsymbol{s}$. Là encore, la visualisation à l'aide de jetons étiquetés par les lettres permet de mieux comprendre les paramètres analysés.

On va considérer qu'une séquence $\boldsymbol{s}$ est une pile de jetons étiquetés et que l'on insère un à un ces jetons dans la structure. Si en insérant un jeton, on s'aperçoit qu'un jeton avec la même lettre est déjà présent, on empile le nouveau jeton sur le premier (voir figure 6.2).

Le coût de traversée cumulé (pour un type de structure donné) associé à un mot $\boldsymbol{s}$ est le coût exigé pour rechercher un à un tous les jetons posés en parcourant $\boldsymbol{s}$ (on s'arrête dès qu'on a trouvé la pile à laquelle appartient le jeton). *Un symbole est donc recherché autant de fois qu'il y a de jetons dans sa pile.* Le coût de recherche simple d'un symbole est simplement le coût pour atteindre la bonne pile. Le coût de recherche cumulé d'un symbole de $\boldsymbol{s}$ est ainsi le coût de recherche simple d'un symbole multiplié par son nombre d'occurrences dans $\boldsymbol{s}$.

Remarque. Dans la suite, l'alphabet est supposé totalement ordonné puisque les listes ordonnées aussi bien que les ABR sont des structures ordonnées. On suppose donc que les symboles de l'alphabet $\mathcal{M} = \{a_1, a_2, \ldots, a_r\}$ vérifient $a_1 < a_2 < \cdots < a_r$.

L'évaluation des espérances des coûts de traversée $\mathrm{E}[\delta]$ correspond à l'analyse (facile) des listes et à l'analyse (plus difficile) des arbres binaires de recherche avec un univers de clés fini et une distribution de probabilité non uniforme (voir par exemple [2, 14, 89] pour des développements de ce type).
Notre approche repose sur une description symbolique de paramètres par des *séries génératrices* et possède sans doute un intérêt intrinsèque indépendant du problème original. C'est pourquoi nous établissons une proposition donnant les valeurs moyennes des coûts de traversée pour les listes ordonnées et les ABR.
Les principes de cette preuve (décomposition d'un langage et transcription aux séries génératrices) sont également utilisés pour déterminer les valeurs moyennes des coûts de traversée cumulés pour les listes ordonnées et les ABR, mais en faisant intervenir l'opération de shuffle. Dans le modèle de Poisson, on a le résultat suivant :

PROPOSITION 6.8. (COÛTS DE TRAVERSÉE CUMULÉE DANS LES LISTES ORDONNÉES ET LES ABR DANS LE MODÈLE DE POISSON). *Soit un ensemble de probabilités $\{p_i\}_{i\in\mathcal{M}}$ associé aux lettres de l'alphabet $\mathcal{M}$ et $\mathcal{P}_z$ le modèle de Poisson de paramètre z pour la taille des mots. Alors, dans le modèle $(\mathcal{P}_z, \{p_i\})$, les espérances des coûts de traversée cumulés pour les listes ordonnées et les arbres binaires de recherche sont respectivement*

$$\mathrm{E}[\delta_L; \mathcal{P}_z, \{p_i\}] = \sum_{j\in\mathcal{M}} P_{[>j]}\; z\;(1-\mathrm{e}^{-p_j z}),$$
$$\mathrm{E}[\delta_B; \mathcal{P}_z, \{p_i\}] = 2\sum_{\substack{(i,j)\in\mathcal{M}^2\\ i<j}} \frac{p_i p_j}{P_{[i,j]}^2}\left[\mathrm{e}^{-zP_{[i,j]}} - 1 + zP_{[i,j]}\right],$$

où $P_{[i,j]} = \sum_{k=i}^{j} p_k$ et $P_{[>j]} = \sum_{k>j} p_k$.

Avant d'écrire la preuve, énonçons le corollaire pour le modèle de Bernoulli obtenu en utilisant un argument de dépoissonisation algébrique (Proposition 6.4).

PROPOSITION 6.9. (COÛTS DE TRAVERSÉE CUMULÉE DANS LES LISTES ORDONNÉES ET LES ABR DANS LE MODÈLE DE BERNOULLI). *Soit un ensemble de probabilités $\{p_i\}_{i\in\mathcal{M}}$ associé aux lettres de l'alphabet $\mathcal{M}$ et $\mathcal{B}_n$ le modèle de Bernoulli à taille fixée n. Alors, dans le modèle $(\mathcal{B}_n, \{p_i\})$, les espérances des coûts de traversée cumulés pour les listes ordonnées et les arbres binaires de recherche sont respectivement*

$$\mathrm{E}[\delta_L; \mathcal{B}_n, \{p_i\}] = \sum_{j\in\mathcal{M}} nP_{[>j]}\;[1-(1-p_i)^{n-1}],$$
$$\mathrm{E}[\delta_B; \mathcal{B}_n, \{p_i\}] = 2\sum_{\substack{(i,j)\in\mathcal{M}^2\\ i<j}} \frac{p_i p_j}{P_{[i,j]}^2}\left[(1-P_{[i,j]})^n - 1 + nP_{[i,j]}\right],$$

où $P_{[i,j]} = \sum_{k=i}^{j} p_k$ et $P_{[>j]} = \sum_{k>j} p_k$.

Preuve. Nous analysons les deux structures l'une après l'autre.
Le paramètre δ_L nécessite l'utilisation de la décomposition de type «shuffle» (' ш '),

$$\mathcal{M}^* = a_1^* \,\text{ш}\, \cdots \,\text{ш}\, a_r^*.$$

Cette décomposition signifie que chaque mot se décompose en sous-mots (éventuellement vides) qui sont des répétitions (a_i^*) du même symbole a_i, mélangés de toutes les façons possibles (à la

manière d'un jeu de carte). On a déjà établi que le produit «shuffle» de deux langages à alphabets disjoints correspond au produit des séries génératrices exponentielles correspondantes. Le coût de recherche d'une clé a_α dans une tranche $\boldsymbol{s}$ est alors égal au nombre de symboles distincts d'indices $i < \alpha$ dans $\boldsymbol{s}$. La série génératrice exponentielle est donc en vertu du théorème 6.1 reliant les langages formels et les séries génératrices.

$$\widehat{f}_\alpha(z, u, x_1, \dots, x_r) = \prod_{i<\alpha} (u(\mathrm{e}^{zx_i} - 1) + 1) \prod_{i \geq \alpha} \mathrm{e}^{zx_i}.$$

Il suffit maintenant de sommer pour tous les symboles $a_\alpha \in \mathcal{M}$ et d'appliquer la formule (6.5) afin d'obtenir $\mathrm{E}[\delta_L]$ dans le modèle de Poisson $\mathcal{P}_z$ avec des probabilités p_i pour les lettres.

Le paramètre δ_B nécessite une approche plus subtile. L'étude classique des ABR considère des arbres binaires de recherche construits sur des permutations tirées aléatoirement de façon uniforme dans $\mathcal{S}_n$ (voir [43]). Nous allons nous inspirer de l'étude classique (qui introduit la notion d'extremum de gauche à droite) pour mener à bien l'analyse. Il faut cependant bien voir que nous autorisons les clés égales. C'est à la fois plus simple car le fait de considérer des mots à la place de permutations va permettre d'utiliser le formalisme reliant les langages formels et les séries génératrices, mais aussi plus compliqué car les notions simples et naturelles sur les permutations doivent être adaptées aux mots et qu'elles perdent peut-être ainsi leur caractère «naturel».

Dans un premier temps, nous décomposons le coût de traversée δ_B selon les lettres de l'alphabet. Pour une tranche $\boldsymbol{s}$, nous considérons le coût de recherche c_α pour atteindre le symbole a_α dans $\mathsf{abr}(\boldsymbol{s})$. Ce coût est la profondeur du nœud a_α dans l'arbre si le nombre d'occurrences $|\boldsymbol{s}|_\alpha$ de la lettre a_α dans $\boldsymbol{s}$ est non nul (on parle de recherche positive).

La preuve comporte trois étapes principales :
– Étude des extrema et des branches extrêmes (recherche des éléments extrêmaux).
– Recherche d'un symbole quelconque. On coupe l'ABR en deux en fonction du symbole cherché afin que le coût de recherche se décompose en coûts de recherche des extrema. Puis, on utilise l'opération de shuffle pour «recomposer» les deux sous-arbres.
– Enfin, on s'attache au problème de la recherche cumulée d'un symbole (tenant compte du nombre d'occurrences du symbole recherché).

Étude des extrema et des branches extrêmes. La première observation clé ici est de remarquer que l'ABR construit sur $\boldsymbol{s}$ est le même que celui construit sur $\boldsymbol{s'}$, où $\boldsymbol{s'}$ est le mot obtenu à partir de $\boldsymbol{s}$ en ne conservant que la première occurrence de chaque symbole (et en respectant l'ordre d'apparition des symboles). En effet seule la première occurrence de chaque symbole entraîne la création d'un nœud. La prochaine occurrence sera simplement «empilée».

Soit a_α la lettre d'indice α de l'alphabet $\mathcal{M}$. Nous introduisons les notations $\boldsymbol{w}_{<\alpha}$, $\boldsymbol{w}_{\leq\alpha}$, $\boldsymbol{w}_{>\alpha}$ et $\boldsymbol{w}_{\geq\alpha}$. Le mot $\boldsymbol{w}_{<\alpha}$ est le mot obtenu à partir de $\boldsymbol{w}$ en supprimant toutes les lettres d'indices strictement inférieurs à α. Les notations $\boldsymbol{w}_{\leq\alpha}$, $\boldsymbol{w}_{>\alpha}$ et $\boldsymbol{w}_{\geq\alpha}$ sont définies de la même manière.

Nous définissons également la notion d'extrema de gauche à droite en prenant en compte les occurrences multiples de symboles.

DÉFINITION 6.10 (EXTREMA STRICTS DE GAUCHE À DROITE). *Soit $\boldsymbol{w} = w_1 w_2 \cdots w_n$ un mot de $\mathcal{M}^*$. La lettre w_j est appelée un* minimum *(resp. maximum) strict de gauche à droite de w si $j = 1$ ou si*

$$(\forall i < j) \quad w_i > w_j \qquad (resp. w_i < w_j).$$

Le nombre de minima stricts (resp. maxima) de gauche à droite pour un mot $\boldsymbol{w}$ est noté $n_{\min}(w)$ (resp. $n_{\max}(w)$).

Remarque. Il est utile de voir dès à présent que les minima (resp. maxima) stricts de gauche à droite d'un mot $\boldsymbol{w}$ correspondent exactement aux nœuds se situant sur la branche extrême gauche (resp. droite) de l'arbre binaire de recherche construit sur w.

Le lemme suivant caractérise le nœud père d'une lettre a_α dont on insère la première occurrence.

LEMME 6.11. *Soit T l'*ABR *construit sur le mot $\boldsymbol{w} = \boldsymbol{\pi} a_\alpha$ où $\boldsymbol{\pi}$ est un mot (éventuellement vide) qui ne contient pas la lettre a_α. Une lettre p est le père de a_α dans l'arbre T si et seulement si $p = \max(\pi_{<\alpha})$ si $a_\alpha > p$ ou $p = \min(\pi_{>\alpha})$ si $a_\alpha < p$.*

Remarque. On confond un nœud avec sa clé.

Preuve. La traversée en ordre symétrique de T donne la liste triée des lettres où selon le cas le père p apparaît soit juste avant soit juste après a_α puisque la lettre a_α correspond à une feuille de l'arbre. □

Tout d'abord, nous étudions les séries génératrices des deux paramètres $n_{\min}$ and $n_{\max}$ (qui sont exactement le nombre de nœuds sur les branches extrêmes gauche et droite).

Il faut analyser la longueur de la branche extrême droite dans un arbre construit sur des mots aléatoires, ou de manière équivalente évaluer le nombre de maxima stricts de gauche à droite. Étant donné l'alphabet $\mathcal{M}$, la décomposition sous forme d'expression régulière

$$\mathcal{M}^* = \prod_{j=1}^{r} (\epsilon + a_j \cdot (a_1 + a_2 + \cdots + a_j)^*)$$

exprime précisément toutes les décompositions possibles des mots selon les maxima stricts de gauche à droite. En effet, on peut toujours décomposer un mot $\boldsymbol{w}$ sous la forme

$$\boldsymbol{w} = \prod_{i=1}^{r} \boldsymbol{m}_i,$$

où le mot $\boldsymbol{m}_i$ ne contient que des symboles d'indices inférieurs ou égaux à j et peut être éventuellement vide. Par exemple sur l'alphabet $\{a, b, c, d\}$, le mot $baadcacba$ s'écrit $(baa)(dcacba)$ (ce qui donne $\boldsymbol{m}_a = \boldsymbol{m}_c = \epsilon$, $\boldsymbol{m}_b = baa$ et $\boldsymbol{m}_d = dcacba$. Un maximum de gauche à droite est la première lettre d'un mot $\boldsymbol{m}_i$.

Appliquant les principes exposés dans le théorème 6.1, on obtient la série génératrice multivariée ordinaire pour le paramètre $n_{\max}$ sur le langage $\mathcal{M}^*$

$$N_{\max}(z, u, x_1, x_2, \ldots, x_r) = \prod_{j=1}^{r} \left(1 + \frac{zux_j}{1 - z(x_1 + \cdots + x_j)}\right).$$

Le coefficient de cette série en $[z^n u^k x_1^{n_1} \cdots x_r^{n_r}]$ est égal au nombre de mots de longueur n ayant k maxima stricts de gauche à droite et n_j occurrences du symbole j. La série génératrice multivariée ordinaire pour les minima s'obtient de façon similaire

$$N_{\min}(z, u, x_1, x_2, \ldots, x_r) = \prod_{j=1}^{r} \left(1 + \frac{zux_j}{1 - z(x_j + \cdots + x_r)}\right).$$

De telles décompositions ont été utilisées par Prodinger [83] pour l'étude du maximum de variables aléatoires distribuées selon une loi géométrique.

Remarque. On peut essayer de donner une vision plus intuitive et peut-être plus formelle. Les arbres binaires de recherche avec les lettres $a_\alpha < a_\beta < a_\gamma$ sur sa branche extrême droite, par exemple, sont décrits grâce à l'expression régulière (utilisant des notations évidentes)

$$\alpha \cdot (\leq \alpha)^* \cdot \beta \cdot (\leq \beta)^* \cdot \gamma (\leq \gamma)^*.$$

La série génératrice multivariée ordinaire est alors

$$\frac{zx_\alpha}{1 - z(x_1 + \cdots + x_\alpha)} \frac{zx_\beta}{1 - z(x_1 + \cdots + x_\beta)} \frac{zx_\gamma}{1 - z(x_1 + \cdots + x_\gamma)}.$$

Afin d'engendrer toutes les branches droites possibles en ordre croissant nous considérons des produits comme

$$\prod_{j=1}^{r} (1 + y_j),$$

où la variable y_j «signifie» que la lettre a_j apparaît sur la branche. Si nous opérons la substitution

$$y_j \mapsto u \frac{zx_j}{1 - z(x_1 + \cdots x_j)},$$

nous obtenons ainsi la fonction génératrice (correspondant au paramètre $n_{\max}$) où u marque le nombre de nœuds sur la branche extrême droite de l'arbre binaire de recherche

$$N_{\max}(z, u, x_1, \ldots, x_r) = \prod_{j=1}^{r} \left(1 + \frac{uzx^j}{1 - z(x_1 + \cdots x_j)}\right) = \prod_{j=1}^{r} \left(\frac{1 - z(x_1 + \cdots x_{j-1}) + (u-1)zx_j}{1 - z(x_1 + \cdots + x_j)}\right).$$

En particulier on a comme il se doit

$$N_{\max}(z, 1, x_1, \ldots, x_r) = \frac{1}{1 - z(x_1 + \cdots + x_r)},$$

puisque nous comptons alors tous les mots quelle que soit la longueur de la branche extrême droite (le produit se «simplifie»).

Coût de recherche simple. La profondeur d'un nœud (son coût de recherche simple) est égale au nombre d'ancêtres sur la branche menant de la racine à ce nœud. Le lemme caractérise les lettres correspondant aux ancêtres de a_α dans un mot terminant par a_α.

LEMME 6.12. *Soit T l'*ABR *obtenu en insérant successivement les lettres du mot $\boldsymbol{w} = \boldsymbol{\pi} a_\alpha$ (où la lettre a_α n'apparaît pas dans le mot $\boldsymbol{\pi}$). La lettre $y \in \boldsymbol{\pi}$ est un ancêtre de a_α dans T si et seulement si la lettre y) est un minimum strict de gauche à droite de $\pi_{>\alpha}$ ou un maximum strict de gauche à droite de $\pi_{<\alpha}$.*

Preuve. Utiliser la définition 6.10 et le lemme 6.11. □

On obtient le corollaire suivant :

COROLLAIRE 6.13. *Soit T l'*ABR *obtenu en insérant successivement les lettres du mot $\boldsymbol{w} = \boldsymbol{\pi} a_\alpha$ (où la lettre a_α n'apparaît pas dans le mot $\boldsymbol{\pi}$). La longueur du chemin de la racine à la lettre a_α dans T est $n_{\min}(\pi_{>\alpha}) + n_{\max}(\pi_{<\alpha})$.*

Ce corollaire donne le principe d'analyse du coût de recherche simple c_α. À partir de l'étude des séries génératrices des deux paramètres $n_{\min}$ and $n_{\max}$ (qui sont exactement le nombre de nœuds sur les branches extrêmes gauche et droite), nous analysons le paramètre c_α en utilisant le produit shuffle.

On considère maintenant le coût de recherche c_α d'un symbole fixé a_α dans un arbre binaire de recherche. La décomposition à l'aide du shuffle (' ш ')

$$(\mathcal{M} \setminus \{a_\alpha\})^* = (a_1 + \cdots + a_{\alpha-1})^* \text{ ш } (a_{\alpha+1} + \cdots + a_r)^*,$$

signifie que chaque mot ne contenant pas la lettre a_α peut être décomposé en deux sous-mots $< a_\alpha$ et $> a_\alpha$, mélangés de toutes les façons possibles. Cette décomposition se calque exactement sur la définition de c_α en fonction de $n_{\min}$ et $n_{\max}$ donnée dans le corollaire. En effet, le paramètre c_α «coût de recherche de a_α» pour un mot $\boldsymbol{w} = \boldsymbol{\pi}(a_\alpha \boldsymbol{v})^*$ (où le mot $\boldsymbol{\pi}$ ne contient pas la lettre a_α) s'écrit $c_\alpha(\mathsf{abr}(\boldsymbol{w})) = n_{\min}(\boldsymbol{\pi}_{>\alpha}) + n_{\max}(\boldsymbol{\pi}_{<\alpha})$. Cela se transpose directement en termes de séries génératrices grâce au shuffle. La série génératrice ordinaire relative au paramètre c_α est

$$C_\alpha(z, u, x_1, \ldots, x_r) = [N_{\max}(z, u, x_1, \ldots, x_{\alpha-1}) \text{ ш } N_{\min}(z, u, x_{\alpha+1}, \ldots, x_r)] \cdot \left[1 + \frac{z x_\alpha}{1 - z(x_1 + \cdots + x_r)}\right], \tag{6.6}$$

où le dernier facteur prend en compte la fin du mot (après que l'on est rencontré la lettre a_α) qui peut éventuellement contenir à nouveau la lettre a_α.

On peut remarquer que l'idée consistant à découper l'arbre binaire de recherche selon le chemin menant à la clé est la même que celle utilisée par l'algorithme classique de coupure pour un abr (algorithme utilisé entre autres pour insérer une clé à la racine).

L'équation (6.6) donne la série génératrice ordinaire qui condense toute l'information, y compris la distribution complète. La série génératrice exponentielle est obtenue en prenant l'inverse par la transformée de Laplace combinatoire. Le coût moyen est obtenu de manière classique en différentiant (6.6) par rapport à la variable u et en posant $u = 1$ selon le principe de (6.5). Le reste des calculs est mené à terme grâce à la transformée de Laplace, en réalisant des décompositions en éléments simples et en utilisant la dérivée logarithmique. Les relations suivantes (évidentes) sont constamment utilisées au cours du calcul

$$\mathcal{L}[\mathrm{e}^{az}] = \frac{1}{1 - az}, \qquad \mathcal{L}[z\mathrm{e}^{az}] = \frac{z}{(1 - az)^2}, \qquad \frac{1}{1 - az} \text{ ш } \frac{1}{1 - bz} = \frac{1}{1 - (a + b)z}.$$

Les calculs sont plus commodes si on introduit les notations

$$X = \sum_{k=1}^{r} x_k, \quad X_{[i,j]} = \sum_{k=i}^{j} x_k, \quad \overline{X_{[i,j]}} = X - X_{[i,j]}, \quad \text{et } \overline{X_\alpha} = \sum_{\substack{k=1 \\ k \neq \alpha}}^{r} x_k.$$

Posons

$$K_\alpha(z, u, x_1, \ldots, x_r) = N_{\min}(z, u, x_1, \ldots, x_{\alpha-1}) \text{ ш } N_{\max}(z, u, x_{\alpha+1}, \ldots, x_r).$$

L'équation (6.6) s'écrit encore

$$C_\alpha(z, u, x_1, \ldots, x_r) = \frac{1 - z\overline{X_\alpha}}{1 - zX} \quad K_\alpha(z, u, x_1, \ldots, x_{\alpha-1}, x_{\alpha+1}, \ldots, x_r),$$

et après quelques calculs, on obtient finalement à la série génératrice cumulée

$$\frac{\partial}{\partial u}\left(C_\alpha(z,u,x_1,\dots,x_r)\right)_{u=1} = \sum_{j=1}^{\alpha-1} \frac{x_j}{X_{[j,\alpha]}}\left(\frac{1}{1-zX} - \frac{1}{1-z\overline{X_{[j,\alpha]}}}\right) + \sum_{j=\alpha+1}^{r} \frac{x_j}{X_{[\alpha,j]}}\left(\frac{1}{1-zX} - \frac{1}{1-z\overline{X_{[\alpha,j]}}}\right).$$

Coût de recherche cumulée. On trouve le coût de recherche cumulé à partir de l'expression de la série génératrice. On s'intéresse non pas au paramètre $c_\alpha(\boldsymbol{w})$ mais au paramètre $|\boldsymbol{w}|_\alpha c_\alpha(\boldsymbol{w})$ qui tient compte du nombre d'occurrences $|\boldsymbol{w}|_\alpha$ du symbole a_α dans $\boldsymbol{w}$. Supposons que nous ayons la série génératrice ordinaire d'un paramètre δ sur le langage L,

$$f(z,u,x_1,\dots,x_r) = \sum_{\boldsymbol{w}\in L} u^{\delta(\boldsymbol{w})} z^{|\boldsymbol{w}|} x_1^{|\boldsymbol{w}|_1}\cdots x_r^{|\boldsymbol{w}|_r},$$

la SGO du paramètre $|\boldsymbol{w}|_\alpha\delta(\boldsymbol{w})$ est

$$x_\alpha \frac{\partial}{\partial x_\alpha} f(z,u,x_1,\dots,x_r).$$

On peut alors utiliser les propositions 6.6 et 6.4 pour obtenir les espérances voulues en sommant pour l'ensemble des lettres de l'alphabet. Ceci termine la preuve de la proposition. □

6.4 L'algorithme Quicksort avec des clés égales

Il peut être judicieux de prendre en compte dans l'algorithme de tri Quicksort le fait qu'il puisse y avoir des clés égales. Du fait du lien très étroit entre l'algorithme et un arbre binaire de recherche qui suit le processus de partitionnement.

L'algorithme Quicksort, introduit par C. A. R. Hoare en 1960 [52], est actuellement reconnu comme la procédure de tri à usage général le plus performant en informatique. L'algorithme possède une riche histoire : beaucoup de modifications ont été proposées afin d'améliorer les performances (voir [90]), les analyses de complexité en temps et en espace ont été menées à bien pour de nombreuses variantes.

Le modèle d'étude «classique» pour Quicksort est celui du modèle des permutations aléatoires, un modèle où toutes les clés sont distinctes. En informatique il arrive fréquemment qu'on ait à trier des données avec des répétitions. Plusieurs questions se posent :
– Quel est l'impact de ces répétitions sur l'algorithme ? [89]
– Comment améliorer Quicksort pour prendre parti de ces répétitions ? [10, 9].
On a déjà parlé du principe de partitionnement avec clés égales dû à Bentley et McIlroy à la section 5.3.4.
Les analyses font intervenir les mêmes constantes que celles de ce chapitre c.-à-d. pour des probabilités d'apparition $\{p_i\}$ de symboles

$$\sum_{i<j} \frac{p_i p_j}{p_i+\cdots+p_j}.$$

Cela semble relié de façon profonde au fait que les clés soient répétées. Cette thèse fournit une approche symbolique pour de tels problèmes. Les preuves calculatoires (basées sur des calculs de récurrences assez complexes) de [89, 14] mériteraient d'être revisitées sous ce nouveau jour.

6.5 Analyse exacte des paramètres de tries

Nous montrons ici que les espérances des principaux paramètres, c.-à-d. la taille, la hauteur et les longueurs de cheminement aussi bien pour les tries «standards» que les versions hybridées, peuvent être exprimés comme des sommes où apparaissent les mesures d'intervalles fondamentaux.

6.5.1 Choix du modèle aléatoire

Les modèle de Poisson et Bernoulli ont été introduits au chapitre 5. Nous précisons ici les liens qui les unissent et pourquoi notre analyse peut être conduite dans le modèle de Poisson, et de quelle façon on peut revenir au modèle de Bernoulli.

Le modèle de Bernoulli considère une suite de n chaînes infinies indépendantes produites par la même source dynamique. Cette suite est de la forme $M(X)$, de cardinal n, et est ainsi obtenue par tirage aléatoire de n points $x_1, x_2, \ldots, x_n$ dans l'intervalle $\mathcal{I}$, munie d'une fonction de densité f (F désigne la fonction de distribution associée). Le modèle de Bernoulli à taille fixée n relatif à une source dynamique probabilisée (S, F) est noté par $(\mathcal{B}_n, \mathsf{S}, F)$.

Plutôt que de fixer le cardinal n de l'ensemble X, il s'avère techniquement préférable de considérer que la suite X possède un nombre variable N d'éléments obéissant à une loi de Poisson de paramètre z

$$\Pr\{N = k\} = e^{-z}\frac{z^k}{k!}.$$

Ce modèle est appelé modèle de Poisson de paramètre z. Lorsqu'il est relatif à une source dynamique probabilisée (S, F), il est noté $(\mathcal{P}_z, \mathsf{S}, F)$. Par définition, les espérances d'une variable aléatoire Y dans les modèles de Poisson et de Bernoulli sont reliés par

$$\mathrm{E}[Y; \mathcal{P}_z, \mathsf{S}, F] = e^{-z}\sum_{n=0}^{\infty}\mathrm{E}[Y; \mathcal{B}_n, \mathsf{S}, F]\frac{z^n}{n!}. \tag{6.7}$$

Une relation similaire permet de représenter les probabilités d'événements qui peuvent toujours être décrits comme les espérances de variables indicatrices. L'intérêt du modèle de Poisson réside dans la complète indépendance entre les événements survenant sur des intervalles disjoints de $\mathcal{I}$. En particulier, le nombre d'éléments qui tombe dans un intervalle de mesure u est lui-même distribué comme une variable de Poisson de paramètre zu. Une telle propriété est vraie notamment pour les intervalles fondamentaux associés à une source (S, F), dont les mesures $u_{\boldsymbol{w}}$ sont données dans l'équation (4.7).

La forte propriété d'indépendance du modèle de Poisson permet d'obtenir facilement les espérances des paramètres de base. Il est alors nécessaire de revenir au modèle plus naturel de Bernoulli. Le processus qui permet de passer d'un modèle de Poisson à un modèle de Bernoulli est appelé dépoissonisation. Plusieurs stratégies de dépoissonisation sont disponibles. Voici celles qui sont utilisées dans cette thèse.

1. *Dépoissonisation algébrique.* Cette technique repose sur le fait que, grâce à l'équation (6.7), une valeur moyenne dans le modèle de Poisson est, à un facteur e^{-z} près, la série génératrice des espérances dans le modèle de Bernoulli

   $$\mathrm{E}[Y; \mathcal{B}_n, \mathsf{S}, F] = n!\,[z^n]\,e^z\,\mathrm{E}[Y; \mathcal{P}_z, \mathsf{S}, F], \tag{6.8}$$

   où $[z^n]h(z)$ représente le coefficient de z^n dans le développement de $h(z)$ en 0. Cette technique est la base de tous les résultats précis dans le modèle de Bernoulli : les théorèmes 6.15, 6.17 sont obtenus de cette façon comme la contrepartie des théorèmes 6.14, 6.16.

2. *Dépoissonisation asymptotique.* Dans le modèle de Poisson, N est fortement concentré autour de sa moyenne z avec une grande probabilité, si bien que le paramètre z joue un rôle tout à fait similaire à celui du paramètre n dans le modèle de Bernoulli. Ces techniques seront présentées plus en détail dans le chapitre qui s'intéresse à l'analyse asymptotique au chapitre 8.

6.5.2 Taille et longueur de cheminement

La forme de la récurrence (5.5), la forme des probabilités à chaque nœud (5.4), et les expressions obtenues dans la proposition 6.8 permettent de dérouler la formule récursive (5.3). En conséquence, les espérances des quatre paramètres additifs δ_S, δ_A, δ_B et δ_L peuvent être exprimées à l'aide des mesures fondamentales.

THÉORÈME 6.14. (ESPÉRANCES DES PARAMÈTRES ADDITIFS DANS LE MODÈLE DE POISSON). *Soit (S, F) une source dynamique probabilisée et $\mathcal{P}_z$ le modèle de Poisson de paramètre z. Alors les espérances dans le modèle $(\mathcal{P}_z, \mathsf{S}, F)$ des paramètres de coût relatifs à la taille d'un trie, à la longueur de cheminement d'un trie-tableau, à la longueur de cheminement d'un trie-liste (avec des listes ordonnées) et à la longueur de cheminement d'un trie-abr sont*

$$\begin{aligned}
\widehat{S}(z) &= \sum_{\boldsymbol{w}\in\mathcal{M}^*} \left[1-(1+zu_{\boldsymbol{w}})e^{-zu_{\boldsymbol{w}}}\right],\\
\widehat{P}_A(z) &= \sum_{\boldsymbol{w}\in\mathcal{M}^*} zu_{\boldsymbol{w}}\left[1-e^{-zu_{\boldsymbol{w}}}\right]\\
\widehat{P}_L(z) &= \sum_{\boldsymbol{w}\in\mathcal{M}^*}\sum_{i\in\mathcal{M}} z\,U_{\boldsymbol{w}\cdot[>i]}(1-e^{-zu_{\boldsymbol{w}\cdot i}})\\
\widehat{P}_B(z) &= 2\sum_{\boldsymbol{w}\in\mathcal{M}^*}\sum_{\substack{(i,j)\in\mathcal{M}^2\\ i<j}} \frac{u_{\boldsymbol{w}\cdot i}u_{\boldsymbol{w}\cdot j}}{U^2_{\boldsymbol{w}\cdot[i,j]}}\left[e^{-zU_{\boldsymbol{w}\cdot[i,j]}}-1+zU_{\boldsymbol{w}\cdot[i,j]}\right],
\end{aligned}$$

où $U_{\boldsymbol{w}\cdot[i,j]} = \sum_{k=i}^{j} u_{\boldsymbol{w}\cdot k}$, et $U_{\boldsymbol{w}\cdot[>j]} = \sum_{k>j} u_{\boldsymbol{w}\cdot k}$.

Nous pouvons maintenant revenir au modèle de Bernoulli grâce aux principes généraux de «dépoissonisation algébrique» résumés dans (6.7) et (6.8) qui traduisent la traduction formelle pour passer des espérances dans le modèle de Poisson à celles dans le modèle de Bernoulli :

$$e^{-az} \mapsto (1-a)^n, \quad ze^{-az} \mapsto n(1-a)^{n-1}.$$

Ainsi les espérances des quatre paramètres additifs dans le modèle de Bernoulli $(\mathcal{B}_n, \mathsf{S}, F)$ peuvent également s'exprimer uniquement en termes de mesures fondamentales. Nous rappelons ici que les deux premiers résultats ont déjà été établis au chapitre 5.

THÉORÈME 6.15. (ESPÉRANCES DES PARAMÈTRES ADDITIFS DANS LE MODÈLE DE BERNOULLI). *Soit $(\mathcal{B}_n, \mathsf{S}, F)$ le modèle de Bernoulli associé à un nombre fixé n de mots émis indépendamment par une source dynamique probabilisée (S, F). Alors les espérances pour la taille d'un trie, la longueur de cheminement d'un trie-tableau, la longueur de cheminement d'un trie-*

liste (avec des listes ordonnées) et la longueur de cheminement d'un trie-abr sont

$$
\begin{aligned}
S(n) &= \sum_{\boldsymbol{w}\in\mathcal{M}^*} \left[1-(1+(n-1)\,u_{\boldsymbol{w}})\,(1-u_{\boldsymbol{w}})^{n-1}\right]\\
P_A(n) &= \sum_{\boldsymbol{w}\in\mathcal{M}^*} nu_{\boldsymbol{w}}[1-(1-u_{\boldsymbol{w}})^{n-1}]\\
P_L(n) &= \sum_{\boldsymbol{w}\in\mathcal{M}^*}\sum_{i\in\mathcal{M}} nU_{\boldsymbol{w}\cdot[>i]}(1-(1-u_{\boldsymbol{w}\cdot i})^{n-1})\\
P_B(n) &= 2\sum_{\boldsymbol{w}\in\mathcal{M}^*}\sum_{\substack{(i,j)\in\mathcal{M}^2\\ i<j}} \frac{u_{\boldsymbol{w}\cdot i}u_{\boldsymbol{w}\cdot j}}{U^2_{\boldsymbol{w}\cdot[i,j]}}\left[(1-U_{\boldsymbol{w}\cdot[i,j]})^n-1+nU_{\boldsymbol{w}\cdot[i,j]}\right],
\end{aligned}
$$

où $U_{\boldsymbol{w}\cdot[i,j]}=\sum_{k=i}^{j}u_{\boldsymbol{w}\cdot k}$ *et* $U_{\boldsymbol{w}\cdot[>j]}=\sum_{k>j}u_{\boldsymbol{w}\cdot k}$ *comme auparavant.*

6.5.3 Hauteur de la structure de trie

Considérons un trie classique (ou de façon équivalente sa version hybridée avec des tableaux) produit par une source dynamique probabilisée (S,F) dans le modèle de Poisson $(\mathcal{P}_z,\mathsf{S},F)$. Un tel trie à une hauteur inférieure ou égale à k si aucun intervalle fondamental de profondeur k ne contient plus d'un mot. La probabilité de cet événement est calculée grâce aux fortes propriétés d'indépendance qu'assurent le modèle de Poisson. À partir de là, on a facilement la valeur moyenne.
Nous rappelons ici des résultats du chapitre 5.

THÉORÈME 6.16 (HAUTEUR DANS LE MODÈLE DE POISSON). *La distribution de la hauteur d'un trie dans le modèle de Poisson* $(\mathcal{P}_z,\mathsf{S},F)$ *est donnée par*

$$\pi_k(z):=\Pr[h\le k]=\prod_{|\boldsymbol{w}|=k}(1+zu_{\boldsymbol{w}})e^{-zu_{\boldsymbol{w}}}=e^{-z}\prod_{|\boldsymbol{w}|=k}(1+zu_{\boldsymbol{w}})$$

et l'espérance de la hauteur est

$$\widehat{H}(z)=\mathrm{E}[h;\mathcal{P}_z,\mathsf{S},F]=\sum_{k=0}^{\infty}[1-\pi_k(z)]\,.$$

Les formules correspondantes dans le modèle de Bernoulli sont obtenues au moyen des principes de «dépoissonisation algébrique» (6.7), (6.8).

THÉORÈME 6.17 (HAUTEUR DANS LE MODÈLE DE BERNOULLI). *Soit* $\pi_{k,n}$ *la probabilité qu'un trie construit sur un* n*-uplet de mots aient une hauteur inférieure ou égale à* k *dans e modèle de Bernoulli* $(\mathcal{B}_n,\mathsf{S},F)$. *La fonction génératrice exponentielle des* $\pi_{k,n}$ *satisfait*

$$\Pi_k(z):=\sum_n \pi_{k,n}\frac{z^n}{n!}=\prod_{|\boldsymbol{w}|=k}(1+zu_{\boldsymbol{w}})\,,$$

est l'espérance correspondante est

$$\mathrm{E}[h;\mathcal{B}_n,\mathsf{S},F]=n![z^n]\sum_{k=0}^{\infty}\left(e^z-\prod_{|\boldsymbol{w}|=k}(1+zu_{\boldsymbol{w}})\right).$$

6.5.4 Séries de Dirichlet associées aux paramètres de trie

Les espérances dans le modèle de Poisson et dans le modèle de Bernoulli se conforment au paradigme des sommes harmoniques de la forme

$$F(z) = \sum_{k\in K} \lambda_k f(\mu_k z),$$

où les quantités $\{\lambda_k\}$, $\{\mu_k\}$ sont appelées amplitudes et fréquences et la fonction f est la fonction de base. L'analyse asymptotique de telles sommes met en jeu classiquement la transformée de Mellin (déjà rencontrée dans la section 8.1). Ainsi la position et la nature des pôles de la série de Dirichlet associée

$$\Lambda(s) := \sum_{k\in K} \lambda_k \mu_k^s$$

constituent l'objet principal d'étude pour obtenir le comportement asymptotique.

Cette approche conduit à considérer plusieurs séries de Dirichlet d'intervalles fondamentaux.

PROPOSITION 6.18 (SÉRIES DE DIRICHLET DE MESURES FONDAMENTALES). *Dans le modèle de Poisson, trois séries de Dirichlet apparaissent pour les tries-tableau, les tries-liste et les tries-*ABR

$$\Lambda^{\langle A\rangle}(F,s) = \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}}^s, \qquad \Lambda^{\langle L\rangle}(F,s) = \sum_{\boldsymbol{w}\in\mathcal{M}^*} \sum_{i\in\mathcal{M}} U_{\boldsymbol{w}\cdot[>i]}\, u_{\boldsymbol{w}\cdot i}^{s-1}, \tag{6.9}$$

$$\Lambda^{\langle B\rangle}(F,s) = 2 \sum_{\boldsymbol{w}\in\mathcal{M}^*} \sum_{\substack{(i,j)\in\mathcal{M}^2\\ i<j}} u_{\boldsymbol{w}\cdot i}\, u_{\boldsymbol{w}\cdot j}\, U_{\boldsymbol{w}\cdot[i,j]}^{s-2}. \tag{6.10}$$

Des formules modifiées peuvent être trouvées dans le modèle de Bernoulli, mais elles ne sont pas directes comme dans le modèle de Poisson. Nous reviendrons sur ce point purement technique au chapitre 8.
Contrairement au cas des paramètres additifs, la quantité exprimant la distribution de probabilité de la hauteur n'est pas une somme harmonique. Cependant, son logarithme l'est,

$$\log \pi_k(z) = -\sum_{|\boldsymbol{w}|=k} z u_{\boldsymbol{w}} \log(1 + z u_{\boldsymbol{w}}),$$

et la série de Dirichlet associée est

$$\Lambda_k^{\langle A\rangle}(F,s) = \sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}}^s \tag{6.11}$$

correspondant aux branches de profondeur k.

L'analyse nécessite alors d'étudier les séries de Dirichlet de mesures fondamentales afin d'obtenir les valeurs moyennes asymptotiques des paramètres. Il faut donc préciser la position des pôles des séries de Dirichlet définies dans les équations (6.9) et (6.10) et aussi déterminer le comportement de la famille de séries de Dirichlet (lorsque k varie) défini dans (6.11). Cela sera fait au moyen de généralisations d'opérateurs de Ruelle qui joueront le rôle «d'opérateurs générateurs» pour les intervalles fondamentaux.

Chapitre 7

Opérateurs de Ruelle généralisés

Sommaire

Ce chapitre explore en détail l'aspect le plus innovant de cette thèse. On pratique afin de pouvoir étudier les séries de Dirichlet des sections précédentes une excursion originale vers l'analyse fonctionnelle. Les *opérateurs générateurs* vont en effet créer le lien entre les sources du chapitre 4 et les séries de Dirichlet du paragraphe précédent. Tout d'abord les propriétés algébriques de ces opérateurs sont présentées. Les multiples séries de Dirichlet à traiter nécessitent l'introduction d'une généralisation de l'opérateur de Ruelle (les opérateurs multisécants) qui on un pouvoir d'expressivité plus grand. Ensuite, les propriétés d'analyse fonctionnelle de ces opérateurs sont examinées. En particulier il faut montrer que les généralisations considérées étendent de façon appropriée l'opérateur de Ruelle de base. Ces opérateurs obéissent tous à une propriété de positivité de type Perron-Frobenius et possèdent donc des objets spectraux dominants. Ce chapitre propose une étude assez fine de ces comportements dominants. Le prochain chapitre (Chapitre 8) utilisera les résultats établis ici sur les propriétés spectrales dominantes pour mener à bien l'analyse asymptotique.

7.1 Opérateurs générateurs : propriétés formelles (algébriques)

Cette section débute par la notion de transformateur de densité (section 7.1.1). Cet opérateur s'étend en un opérateur de transfert grâce à l'introduction d'un paramètre complexe s supplémentaire (section 7.1.2). Nous proposons alors une classe d'opérateurs de Ruelle généralisés, les opérateurs multi-sécants (section 7.1.3), où les dérivées apparaissant dans les opérateurs de transfert sont remplacées par des fonctions sécantes. L'opérateur agit dès lors sur des fonctions multivariées. Les opérateurs construits sur des multi-sécantes permettent d'exprimer un grand nombre de séries de Dirichlet d'intervalles fondamentaux, tout particulièrement celles qui apparaissent au cours de l'analyse des paramètres des tries (sections 7.1.5, 7.1.6).

7.1.1 Transformateurs de densité.

Il y a une relation directe entre la dynamique de la source S, les réponses aux principales questions soulevées dans la section 6.5.4, et les propriétés spectrales d'un opérateur transformant les fonctions de distribution de probabilité. L'ingrédient de base, qui a été étudié dans la théorie des systèmes dynamiques, est la classe des *opérateurs de transfert* [5, 87]. Dans sa forme la plus simple, l'opérateur de transfert associé à un système dynamique de base est «le transformateur de densité»,

$$\mathcal{G}[f](x) := \sum_{i\in\mathcal{M}} |h_i'(x)|\ f\circ h_i(x).$$

Le terme le désignant provient de ses propriétés évidentes : si X est une variable aléatoire avec une fonction de densité f, alors la densité de $T(X)$ est $\mathcal{G}[f]$. En d'autres termes, l'opérateur $\mathcal{G}$ décrit une étape du processus lié à la source. L'opérateur composante correspondant au i^{e} terme est noté $\mathcal{G}_{[i]}$; il est défini par

$$\mathcal{G}_{[i]}[f](x) := |h_i'(x)|\ f\circ h_i(x). \tag{7.1}$$

Ainsi on a

$$\mathcal{G} = \sum_{i\in\mathcal{M}} \mathcal{G}_{[i]}. \tag{7.2}$$

De la même façon, un «transformateur de densité» associé à un système dynamique de Markov peut être défini. Il y a maintenant r fonctions de densité $(f_1, f_2, \ldots, f_r)$ qui correspondent à des «densités conditionnelles» : $f_j(x)$ est la fonction de densité au point x quand le dernier symbole émis est j. Partant d'une densité initiale f, et, après une itération de la fonction de décalage associée au système initial S_0, on obtient

$$f_j(x) = |h_{j|0}'(x)|\ f\circ h_{j|0}(x).$$

Plus généralement, la suite de densités conditionnelles $(f_1, f_2, \ldots, f_r)$ à une étape de processus, et la suite de densités conditionnelles à l'étape suivante $(g_1, g_2, \ldots, g_r)$ sont reliées par une matrice d'opérateurs $\mathcal{G}$ qui est construite d'après les transformateurs de densité $\mathcal{G}_j$ associés à chaque système dynamique S_j. Le transformateur de densité $\mathcal{G}_j$ associé à S_j agit sur f_j

$$\mathcal{G}_j[f_j](x) := \sum_{i\in\mathcal{M}} |h_{i|j}'(x)|\ f_j\circ h_{i|j}(x).$$

Chaque terme de la somme précédente définit un opérateur qui sera noté $\mathcal{G}_{[i|j]}$,

$$\mathcal{G}_{[i|j]}[f](x) := |h'_{i|j}(x)|\ f \circ h_{i|j}(x), \tag{7.3}$$

et chaque terme $\mathcal{G}_{[i|j]}[f_j]$ représente la «part» de la nouvelle densité g_i qui «provient» de la densité f_j. Nous considérons maintenant la matrice à r lignes et r colonnes $\mathcal{G}$ dont le coefficient général est $\mathcal{G}_{[i|j]}$. Il s'agit de la matrice

$$\mathcal{G} = \left(\mathcal{G}_{[i|j]}\right), \tag{7.4}$$

(i est l'indice pour la ligne, et j l'indice pour la colonne). Cette matrice $\mathcal{G}$ est en fait le transformateur de densité, puisqu'elle transforme la suite de densités conditionnelles $(f_1, f_2, \ldots, f_r)$ à une étape donnée du processus itératif, en la suite de densités conditionnelles $(g_1, g_2, \ldots, g_r)$ à l'étape suivante :

$$\begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_r \end{pmatrix} = \left(\mathcal{G}_{[i|j]}\right) \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_r \end{pmatrix}.$$

7.1.2 Opérateurs de Ruelle classiques

Dans tous les cas, que ce soit pour un système dynamique de base ou markovien, il s'avère extrêmement utile de travailler avec des opérateurs plus généraux, appelés *opérateurs de Ruelle*. Chaque opérateur composante dans (7.1) ou (7.3) dépend maintenant d'un paramètre complexe s qui s'interprète dans le cadre de la physique statistique comme la *température*. Cet opérateur est défini à l'aide de l'extension analytique $\widetilde{h}_i$ de $|h'_i|$. Les nouveaux opérateurs composantes sont notés respectivement $\mathcal{G}_{s,[i]}$ et $\mathcal{G}_{s,[i|j]}$

$$\mathcal{G}_{s,[i]}[f](z) := \widetilde{h}_i(z)^s\ f \circ h_i(z) \tag{7.5}$$

$$\mathcal{G}_{s,[i|j]}[f](z) := \widetilde{h}_{i|j}(z)^s\ f \circ h_{i|j}(z). \tag{7.6}$$

De même que dans (7.2) ou (7.4), les opérateurs de Ruelle sont maintenant définis par

$$\mathcal{G}_s = \sum_{i=1}^{r} \mathcal{G}_{s,[i]}, \qquad \mathcal{G}_s := \left(\mathcal{G}_{s,[i|j]}\right) \tag{7.7}$$

dans le cas de base et dans le cas markovien. La dynamique du processus est *a priori* décrite pour une valeur du paramètre $s = 1$ (i.e., $\mathcal{G} = \mathcal{G}_1$), mais beaucoup d'autres propriétés sont reliées à des valeurs complexes de s différentes de 1.

7.1.3 Opérateurs de Ruelle généralisés

Dans [100, 102], Brigitte Vallée a introduit un nouvel outil, *l'opérateur de Ruelle généralisé*, utilisant des *sécantes* de branches inverses plutôt que des *tangentes* $|h'(z)|$. L'objet de notre étude, les tries, nous incite à construire une généralisation plus large encore qui utilise des *multi-sécantes*. En effet, on s'intéresse à des séries de Dirichlet d'intervalles fondamentaux, et on verra que ces opérateurs permettent en quelque sorte de les «engendrer». L'opérateur hyper-généralisé obtenu, $\mathfrak{G}_s$, agit alors sur un espace de fonctions de m variables complexes. Cet opérateur constitue alors

une extension de degré m de l'opérateur de Ruelle. Dans cette thèse, nous aurons besoin d'extensions de degrés $m = 1, 2, 3$ ou 4.

Il y a deux façons d'étendre une branche inverse h en une fonction de m variables. La première, qui ne dépend que du degré m de l'extension, est

$$V_m[h](x_1, \dots, x_m) = (h(x_1), \dots, h(x_m)).$$

La seconde, notée $H_s[h]$, doit satisfaire les trois propriétés suivantes,

(i) H_s étend la fonction tangente $\widetilde{h}(z)^s$ et sa restriction sur la diagonale coïncide avec la fonction tangente

$$H_s[h](\underbrace{x, x, \dots, x}_{m \text{ fois}}) = \widetilde{h}^s(x) \text{ pour tout } m \geq 1.$$

(ii) H_s est multiplicative[1] dans le sens suivant

$$H_s[h \circ g](x_1, x_2, \dots, x_m) = H_s[h]\left(g(x_1), g(x_2), \dots, g(x_m)\right) \times H_s[g](x_1, x_2, \dots, x_m).$$

(iii) Lorsque le paramètre s et les points x_i sont tous réels, H_s est positive

$$H_s[h](x_1, x_2, \dots, x_m) > 0.$$

Certains exemples de telles fonctions H_s sont particulièrement importantes pour notre analyse : ce sont les applications que nous appelons multi-sécantes définies à partir d'un ensemble de fonctions affines $s \mapsto d_{i,j}(s)$

$$H_s[h](x_1, \dots, x_m) := \prod_{1 \leq i,j \leq m} \left| \frac{h(x_i) - h(x_j)}{x_i - x_j} \right|^{d_{i,j}(s)} \text{ avec } \sum_{1 \leq i,j \leq m} d_{i,j}(s) = s. \tag{7.8}$$

Cette définition étend la notion de fonction sécante $G(u, v)$ d'une fonction $g(x)$ définie par $G(u, v) := \left| \frac{g(u) - g(v)}{u - v} \right|$. Les propriétés $(i), (ii), (iii)$ sont alors clairement vérifiées par les multi-sécantes ainsi définies.

Chaque opérateur composante dans (7.5) ou (7.6) est maintenant défini grâce à l'extension analytique $\widetilde{H}_s[h_i]$ de la multi-sécante $H_s[h_i]$ relative à la branche inverse h_i. Ces nouveaux opérateurs composantes sont notés $\mathfrak{G}_{s,[i]}$ ou $\mathfrak{G}_{s,[i|j]}$ respectivement et agissent maintenant sur des fonctions F de m variables complexes de la façon suivante

$$\mathfrak{G}_{s,[i]}[F] := \widetilde{H}_s[h_i]\, F \circ V_m[h_i], \qquad \mathfrak{G}_{s,[i|j]}[F] := \widetilde{H}_s[h_{i|j}]\, F \circ V_m[h_{i|j}].$$

De manière analogue à (7.7), les opérateurs de Ruelle généralisés sont définis par

$$\mathfrak{G}_s := \sum_{i=1}^{r} \mathfrak{G}_{s,[i]}, \qquad \mathfrak{G}_s := \left(\mathfrak{G}_{s,[i|j]}\right)$$

respectivement dans le cas de base et dans le cas markovien.

[1] On peut encore écrire plus succinctement cette propriété de multiplicativité grâce à la notation $V_m[h]$ (introduite ci-dessus)

$$H_s[h \circ g] = H_s[h] \circ V_m[g] \times H_s[g].$$

Ces généralisations de l'opérateur de Ruelle constituent effectivement des extensions de l'opérateur de Ruelle classique dans le sens suivant : Si f est la fonction diagonale de F, i.e., $f(u) := F(u, \ldots, u)$ (m fois), la relation suivante est vraie sur la diagonale $x_1 = \cdots = x_m = u$

$$\mathfrak{G}_s[F](\underbrace{u, \ldots, u}_{m \text{ fois}}) = \mathcal{G}_s[f](u). \tag{7.9}$$

Le cas de la dimension $m = 1$ correspond exactement à l'opérateur classique de Ruelle $\mathcal{G}_s$.

7.1.4 Les trois opérateurs généralisés relatifs aux tries

D'après la section 6.5.4 et les équations (6.9), (6.10), (6.11), les séries de Dirichlet émergeant de l'analyse des tries sont

$$\Lambda^{\langle A\rangle}(F,s) = \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}}^s, \qquad \Lambda_k^{\langle A\rangle}(F,s) = \sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}}^s$$
$$\Lambda^{\langle B\rangle}(F,s) = 2\sum_{\boldsymbol{w}\in\mathcal{M}^*} \sum_{\substack{(i,j)\in\mathcal{M}^2\\ i<j}} u_{\boldsymbol{w}\cdot i}\, u_{\boldsymbol{w}\cdot j}\, U_{\boldsymbol{w}\cdot[i,j]}^{s-2},$$
$$\Lambda^{\langle L\rangle}(F,s) = \sum_{\boldsymbol{w}\in\mathcal{M}^*} \sum_{i\in\mathcal{M}} U_{\boldsymbol{w}\cdot[>i]}\, u_{\boldsymbol{w}\cdot i}^{s-1}.$$

Les deux premières séries correspondent aux séries de mesures fondamentales définies dans la section 4 et elles jouent d'ores et déjà un rôle central dans un grand nombre d'analyses portant sur des problèmes liés à des préfixes de mots ; voir [102].

Chaque terme de la série de Dirichlet nécessite dans les deux premières séries (qui font intervenir le même opérateur) deux points, qui sont les extrémités de l'intervalle fondamental, trois points dans le cas des tries-liste ou quatre points dans le cas des tries-abr. En conséquence, les opérateurs de Ruelle agissent sur des fonctions de deux variables ($m = 2$), dans le premier cas, de trois variables ($m = 3$) ou de quatre variables ($m = 4$) pour les cas des tries hybrides. Nous les noterons respectivement $\mathfrak{G}_s^{\langle A\rangle}$, $\mathfrak{G}_s^{\langle L\rangle}$ et $\mathfrak{G}_s^{\langle B\rangle}$.

La proposition suivante (qui tient plus de la définition en fait) explicite les opérateurs généralisés qui seront utilisés dans la suite pour engendrer les intervalles fondamentaux.

PROPOSITION 7.1 (OPÉRATEURS GÉNÉRALISÉS POUR LES TRIES). *Les opérateurs généralisés utilisés pour les tries $\mathfrak{G}_s^{\langle A\rangle}$, $\mathfrak{G}_s^{\langle L\rangle}$,$\mathfrak{G}_s^{\langle B\rangle}$ sont définis à l'aide des multi-sécantes*

$$H_s^{\langle A\rangle}[h](x_1,x_2) = \left|\frac{h(x_1)-h(x_2)}{x_1-x_2}\right|^s \tag{7.10}$$
$$H_s^{\langle L\rangle}[h](x_1,x_2,x_3) = \left|\frac{h(x_2)-h(x_3)}{x_2-x_3}\right| \cdot \left|\frac{h(x_1)-h(x_2)}{x_1-x_2}\right|^{s-1} \tag{7.11}$$
$$H_s^{\langle B\rangle}[h](x_1,x_2,x_3,x_4) = \left|\frac{h(x_1)-h(x_2)}{x_1-x_2}\right| \cdot \left|\frac{h(x_3)-h(x_4)}{x_3-x_4}\right| \cdot \left|\frac{h(x_1)-h(x_4)}{x_1-x_4}\right|^{s-2}. \tag{7.12}$$

Ainsi L'opérateur $\mathfrak{G}_s^{\langle A\rangle}$ appliqué à une fonction F au point (x_1, x_2) s'écrit

$$\mathfrak{G}_s^{\langle A\rangle}[F](x_1,x_2) = \sum_{i\in\mathcal{M}} \left|\frac{h_i(x_1)-h_i(x_2)}{x_1-x_2}\right|^s F(h_i(x_1), h_i(x_2)).$$

Les opérateurs associés dans le cas des tries hybrides sont

$$\mathfrak{G}_s^{\langle B\rangle}[F] = \sum_{i\in\mathcal{M}} H_s^{\langle B\rangle}[h_i]\ F\circ V_4[h_i], \qquad \mathfrak{G}_s^{\langle L\rangle}[F] = \sum_{i\in\mathcal{M}} H_s^{\langle L\rangle}[h_i]\ F\circ V_3[h_i].$$

Ces opérateurs sont définis en calquant leur structure sur la série des mesures d'intervalles fondamentaux mis en jeu. C'est l'objet des deux parties qui suivent d'expliciter le lien entre ces opérateurs et les séries. Nous allons montrer comment les opérateurs $\mathfrak{G}_s$ engendrent toutes les branches inverses $h_{\boldsymbol{w}}$ de profondeur quelconque, d'abord dans le cas d'un système dynamique de base (Partie 7.1.5), puis dans le cas markovien (Partie 7.1.6).

7.1.5 Séries de Dirichlet : cas de base

Du fait de la propriété de multiplicativité (ii), le k^{e} itéré de $\mathfrak{G}_s$ fait intervenir toutes les branches inverses $h_{\boldsymbol{w}}$ de profondeur k,

$$\mathfrak{G}_s^k[F] = \sum_{\boldsymbol{w}\in\mathcal{M}^k} \widetilde{H}_s[h_{\boldsymbol{w}}]\ F\circ V_m[h_{\boldsymbol{w}}], \tag{7.13}$$

où la fonction $\widetilde{H}_s[h_{\boldsymbol{w}}]$ est l'extension analytique de la multi-sécante $H_s[h_{\boldsymbol{w}}]$.
De même, le quasi-inverse $(I-\mathfrak{G}_s)^{-1}$, défini comme la somme formelle de tous les itérés de l'opérateur, représente alors *toutes* les itérations possibles de l'opérateur $\mathfrak{G}_s$, et s'exprime sous la forme d'une somme sur toutes les branches inverses

$$(I-\mathfrak{G}_s)^{-1}[F] = \sum_{\boldsymbol{w}\in\mathcal{M}^*} \widetilde{H}_s[h_{\boldsymbol{w}}]\ F\circ V_m[h_{\boldsymbol{w}}].$$

Elles sont engendrées par l'extension de degré $m=2$ de l'opérateur utilisant la «vraie» sécante définie en (7.10). appliquée à la fonction $L_s^{\langle A\rangle} := H_s^{\langle A\rangle}[F]$ qui est la sécante de la distribution initiale F. Plus précisément, on a

PROPOSITION 7.2. (SÉRIE DE DIRICHLET DES MESURES FONDAMENTALES, CAS DE BASE). *Les séries de Dirichlet pour les mesures fondamentales respectivement de profondeur k et de profondeur quelconque s'expriment à l'aide de l'opérateur $\mathfrak{G}_s^{\langle A\rangle}$*

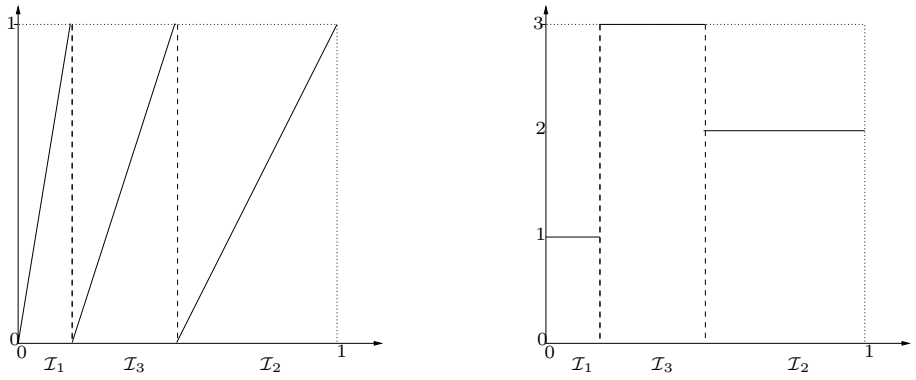
$$\begin{aligned}\Lambda_k^{\langle A\rangle}(F,s) &:= \sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}}^s = \left(\mathfrak{G}_s^{\langle A\rangle}\right)^k [L_s^{\langle A\rangle}](0,1), \\ \Lambda^{\langle A\rangle}(F,s) &:= \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}}^s = (I-\mathfrak{G}_s^{\langle A\rangle})^{-1}[L_s^{\langle A\rangle}](0,1).\end{aligned} \tag{7.14}$$

Preuve. Il suffit de remarquer que

$$u_{\boldsymbol{w}} = |F(h_{\boldsymbol{w}}(0)) - F(h_{\boldsymbol{w}}(1))|^s.$$

□

Les deux autres séries, $\Lambda^{\langle L\rangle}$, $\Lambda^{\langle B\rangle}$, correspondent aux tries hybrides. Contrairement au cas précédent, l'ordre des symboles devient important puisque l'on considère alors à la fois un intervalle fondamental et ses subdivisions. En effet, l'ordre sur les symboles de l'alphabet, qui permet d'obtenir un ordre lexicographique sur les mots, ne coïncide pas nécessairement avec l'ordre naturel sur la partition (topologique) $\{\mathcal{I}_m\}$. Nous supposons ici que ces ordres sont «compatibles», c.-à-d.



(i) Application de décalage T (ii) Application de codage σ

FIG. 7.1 – Exemple de mécanisme de source sans mémoire associée aux probabilités $(\frac{1}{6}, \frac{1}{3}, \frac{1}{2})$ illustrant le cas où la partition topologique $\{\mathcal{I}_m\}$ n'est pas ordonnée.

qu'ils sont soit les mêmes, soit inverse l'un de l'autre. Ainsi, la fonction de codage σ est soit croissante, soit décroissante et la réunion de tout sous-ensemble d'intervalles fondamentaux (de même hauteur) correspondant à des branches successives constitue bien un intervalle. Un exemple de mécanisme de source pour lesquels on ne peut pas engendrer $\Lambda^{\langle L\rangle}$ et $\Lambda^{\langle B\rangle}$ grâce aux opérateurs qui vont être introduits ici, est représenté sur la figure 7.1.

Les fonctions particulières utilisées pour engendrer les séries de Dirichlet sont les multi-sécantes de la distribution initiale F,

$$L_s^{\langle L\rangle} = H_s^{\langle L\rangle}[F], \qquad L_s^{\langle B\rangle} = H_s^{\langle B\rangle}[F],$$

où $H_s^{\langle B\rangle}$ et $H^{\langle L\rangle}$ sont définis dans les équations (7.11) et (7.12).

Cas du codage croissant. Supposons dans un premier temps que le codage soit croissant (σ est croissante). Nous introduisons pour une branche inverse $h_{\boldsymbol{w}}$ la notation

$$(h_{\boldsymbol{w}}^-, h_{\boldsymbol{w}}^+) = (\inf(h_{\boldsymbol{w}}(0), h_{\boldsymbol{w}}(1)), \sup(h_{\boldsymbol{w}}(0), h_{\boldsymbol{w}}(1))).$$

de telle sorte que l'intervalle fondamental $\mathcal{I}_{\boldsymbol{w}}$ a pour extrémités $h_{\boldsymbol{w}}^-$ et $h_{\boldsymbol{w}}^+$. La mesure de l'intervalle fondamental (appelée mesure fondamentale) est $u_{\boldsymbol{w}} = |F(h_{\boldsymbol{w}}^-) - F(h_{\boldsymbol{w}}^+)|$ où F est la fonction de distribution initiale.
On a pour les branches inverses de profondeur 1 (avec un alphabet fini pour l'instant[2]),

$$0 = h_1^- < h_1^+ = h_2^- < h_2^+ = h_3^- < \cdots < h_{r-1}^+ = h_r^- < h_r^+ = 1 \tag{7.15}$$

puisque les intervalles $\mathcal{I}_m$ sont en ordre croissant. On a également

$$u_{\boldsymbol{w}\cdot i} = |F \circ h_{\boldsymbol{w}}(h_i^-) - F \circ h_{\boldsymbol{w}}(h_i^+)| \text{ et } U_{\boldsymbol{w}\cdot[i,j]} = |F \circ h_{\boldsymbol{w}}(h_i^-) - F \circ h_{\boldsymbol{w}}(h_j^+)|.$$

[2]Le cas de l'alphabet fini est aisément transposable au cas de l'alphabet infini en écrivant formellement $r = \infty$ dans les formules finales et en posant $h_\infty^+ = 1$.

L'opérateur du trie-abr s'exprime en fonction des opérateurs composantes

$$\begin{aligned}\mathfrak{G}^{\langle B\rangle}_{s,[\boldsymbol{w}]}[L^{\langle B\rangle}_s](x_1,x_2,x_3,x_4) &= H^{\langle B\rangle}_s[h_{\boldsymbol{w}}](x_1,x_2,x_3,x_4)\times L^{\langle B\rangle}_s\circ V_4[h_{\boldsymbol{w}}](x_1,x_2,x_3,x_4)\\ &= \left|\frac{F\circ h_{\boldsymbol{w}}(x_1)-F\circ h_{\boldsymbol{w}}(x_2)}{x_1-x_2}\right|\left|\frac{F\circ h_{\boldsymbol{w}}(x_3)-F\circ h_{\boldsymbol{w}}(x_4)}{x_3-x_4}\right|\\ &\qquad \left|\frac{F\circ h_{\boldsymbol{w}}(x_1)-F\circ h_{\boldsymbol{w}}(x_4)}{x_1-x_4}\right|^{s-2}.\end{aligned}$$

Ainsi un terme de la série de Dirichlet $\Lambda^{\langle B\rangle}(F,s)$ s'écrit

$$u_{\boldsymbol{w}\cdot i}\,u_{\boldsymbol{w}\cdot j}\,U^{s-2}_{\boldsymbol{w}\cdot[i,j]} = u^\star_i u^\star_j {U^\star_{[i,j]}}^{s-2}\mathfrak{G}^{\langle B\rangle}_{s,[\boldsymbol{w}]}[L^{\langle B\rangle}_s](h^-_i,h^+_i,h^-_j,h^+_j),$$

où $u^\star_i=|h^-_i-h^+_i|$, $u^\star_j=|h^-_j-h^+_j|$, $U^\star_{[i,j]}=|h^-_i-h^+_j|$ font référence aux mesures canoniques des intervalles fondamentaux de profondeur 1.

Cas du codage décroissant. Il suffit d'adapter ce qui précède en tenant compte du fait que les intervalles disjoints $\mathcal{I}_m$ sont ordonnés de façon décroissante sur $[0,1]$. Les formules précédentes restent valides si l'on définit symétriquement

$$(h^-,h^+)=(\sup(h(0),h(1)),\inf(h(0),h(1))).$$

On a l'analogue de l'équation (7.15)

$$1=h^-_1>h^+_1=h^-_2>h^+_2=h^-_3>\cdots>h^+_{r-1}=h^-_r>h^+_r=0.$$

Avec cette définition de (h^-,h^+) les expressions des séries de Dirichlet sont alors exactement les mêmes que dans le cas du codage croissant.
Finalement, en définissant le couple (h^-,h^+) comme

$$(h^-,h^+)=\begin{cases}(\inf(h(0),h(1)),\sup(h(0),h(1))) & \text{si le codage } \sigma \text{ est croissant,}\\ (\sup(h(0),h(1)),\inf(h(0),h(1))) & \text{si le codage } \sigma \text{ est décroissant,}\end{cases} \tag{7.16}$$

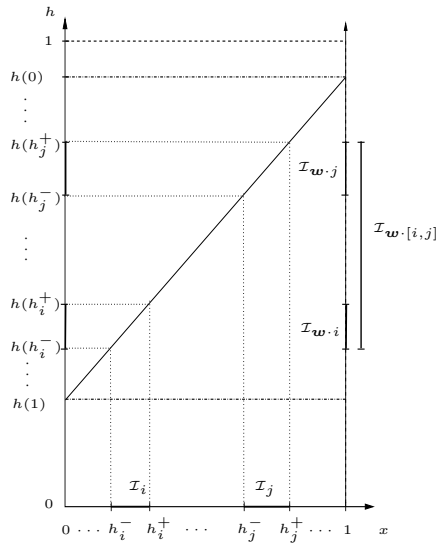
on obtient une seule et unique formule pour la mesure $U_{\boldsymbol{w}\cdot[i,j]}=|F\circ h_{\boldsymbol{w}}(h^-_i)-F\circ h_{\boldsymbol{w}}(h^+_j)|$ (voir la figure 7.2 pour une vision plus intuitive de ce fait) ce qui permet d'établir la proposition suivante.

PROPOSITION 7.3. (EXPRESSION DES SÉRIES DE DIRICHLET POUR LES TRIES HYBRIDES, CAS DE BASE). *Prenant la notation (7.16), pour une source (S,F) avec un codage σ monotone, les séries de Dirichlet s'expriment sous la forme*

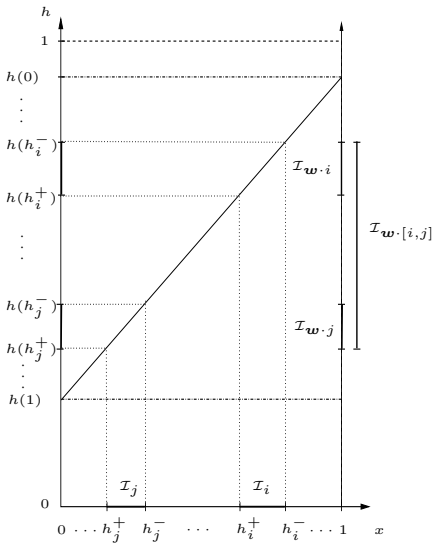
$$\Lambda^{\langle B\rangle}(F,s)=2\sum_{\substack{(i,j)\in\mathcal{M}^2\\ i<j}} u^\star_i u^\star_j {U^\star_{[i,j]}}^{s-2}(I-\mathfrak{G}^{\langle B\rangle}_s)^{-1}[L^{\langle B\rangle}_s](h^-_i,h^+_i,h^-_j,h^+_j), \tag{7.17}$$

$$\Lambda^{\langle L\rangle}(F,s)=\sum_{i\in\mathcal{M}} {u^\star_i}^{s-1}\,U^\star_{[>i]}(I-\mathfrak{G}^{\langle L\rangle}_s)^{-1}[L^{\langle L\rangle}_s](h^-_i,h^+_i,h_r+), \tag{7.18}$$

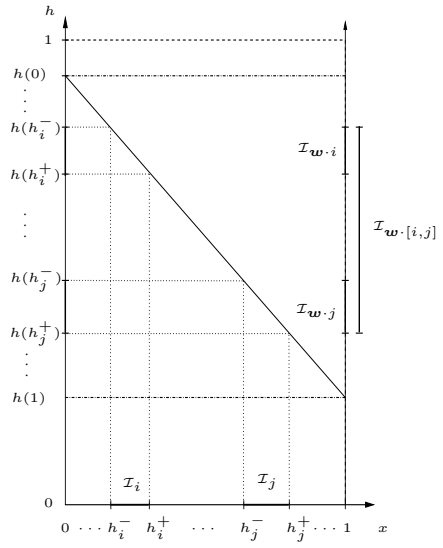
où dans le cas de l'alphabet infini, on remplace formellement $r=\infty$. Que le cardinal de l'alphabet r soit fini ou non, h^+_r vaut 1 ou 0 selon que le codage est croissant ou décroissant. Les quantités $u^\star_i=|h^-_i-h^+_i|$, $U^\star_{[>i]}=|h^+_i-h^+_r|$ et $U^\star_{[i,j]}=|h^-_i-h^+_j|$ font référence aux mesures canoniques des intervalles fondamentaux de profondeur 1.



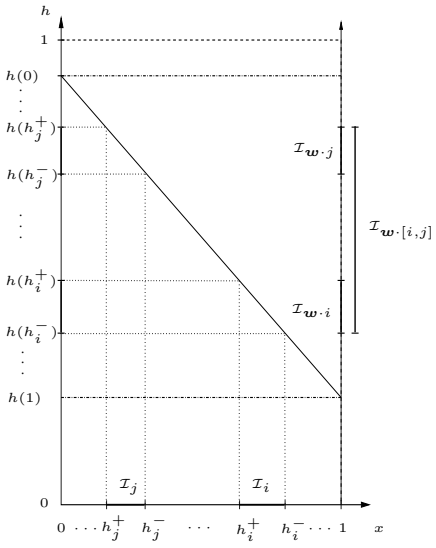
(i) Codage σ croissant & branche inverse h croissante



(ii) Codage σ décroissant & branche inverse h croissante



(iii) Codage σ croissant & branche inverse h décroissante



(iv) Codage σ décroissant & branche inverse h décroissante

FIG. 7.2 – Les quatre configurations possibles par rapport au sens de variation de la branche inverse h considérée (associée au préfixe $\boldsymbol{w}$) et par rapport au sens du codage pour le calcul de l'intervalle $\mathcal{I}_{\boldsymbol{w}\cdot[i,j]}$.

7.1.6 Séries de Dirichlet : cas markovien

Dans le cas markovien, le coefficient (i,j) de la k^{e} itérée de la matrice $\mathfrak{G}_s$ met en jeu toutes les branches h relatives aux mots $\boldsymbol{w} = (m_1, \ldots, m_k)$ qui commencent par j ($m_1 = j$) et finissent par i ($m_k = i$).

Pour les séries de Dirichlet liées aux tries-tableau, nous désirons engendrer toutes les branches inverses de profondeur k. Nous considérons dans un premier temps l'opérateur $\mathfrak{M}_s$ correspondant au système dynamique initial S_0

$$\mathfrak{M}_s[F] := \begin{pmatrix} \mathfrak{G}_{s,[1|0]} \\ \mathfrak{G}_{s,[2|0]} \\ \vdots \\ \mathfrak{G}_{s,[i|0]} \\ \vdots \\ \mathfrak{G}_{s,[r|0]} \end{pmatrix} [F].$$

Si $\mathbf{e}$ désigne le vecteur unité de dimension r, i.e., ${}^t\mathbf{e} = (1, 1, \ldots 1)$ (r fois), alors ${}^t\mathbf{e}\, \mathfrak{M}_s$ désigne l'opérateur de Ruelle généralisé associé à la source S_0. Pour $k \geq 1$, les opérateurs ${}^t\mathbf{e}\, \mathfrak{G}_s^{k-1}\, \mathfrak{M}_s$ engendrent toutes les branches inverses de profondeur k et nous obtenons l'analogue de (7.13)

$$\begin{aligned} {}^t\mathbf{e}\, \mathfrak{G}_s^{k-1}\, \mathfrak{M}_s\, [F] &= \sum_{\boldsymbol{w} \in \mathcal{M}^k} \widetilde{H}_s[h_{\boldsymbol{w}}]\, F \circ V_m[h_{\boldsymbol{w}}], \\ \Big[I + {}^t\mathbf{e}\, (I - \mathfrak{G}_s)^{-1}\, \mathfrak{M}_s \Big]\, [F] &= \sum_{\boldsymbol{w} \in \mathcal{M}^*} \widetilde{H}_s[h_{\boldsymbol{w}}]\, F \circ V_m[h_{\boldsymbol{w}}]. \end{aligned}$$

Les formules pour les séries de Dirichlet associées aux mesures fondamentales sont rassemblées dans la proposition suivante

PROPOSITION 7.4. (SÉRIE DE DIRICHLET DES MESURES FONDAMENTALES, CAS MARKOVIEN). *Les séries de Dirichlet pour les mesures fondamentales respectivement de profondeur k et de profondeur quelconque s'expriment à l'aide des opérateurs $\mathfrak{G}_s^{\langle A \rangle}$ et $\mathfrak{M}_s$*

$$\Lambda^{\langle A \rangle}(F, s) = \left[I + {}^t\mathbf{e} \left(I - \mathfrak{G}_s^{\langle A \rangle} \right)^{-1} \mathfrak{M}_s \right] [L_s^{\langle A \rangle}](0, 1), \tag{7.19}$$

$$\Lambda_k^{\langle A \rangle}(F, s) = {}^t\mathbf{e} \left(\mathfrak{G}_s^{\langle A \rangle} \right)^{k-1} \mathfrak{M}_s\, [L_s^{\langle A \rangle}](0, 1) \tag{7.20}$$

De même, si $\mathbf{e}_\ell$ est le ℓ^{e} vecteur de la base canonique, l'opérateur

$${}^t\mathbf{e}_\ell\, \mathfrak{G}_s^{k-1}\, \mathfrak{M}_s$$

engendre toutes les branches inverses de profondeur k relatives aux préfixes $\boldsymbol{w}$ qui finissent par le symbole ℓ. Reprenant la notation (7.16), on définit $(h^+_{i|\ell}, h^-_{i|\ell})$ selon que le codage σ_ℓ relatif au système dynamique S_ℓ est croissant ou non. On voit apparaître dans les expressions des séries de Dirichlet associées aux tries-liste et tries-abr les mesures fondamentales canoniques étendues au cas markovien

$$u^\star_{i|\ell} := |h^-_{i|\ell} - h^+_{i|\ell}|,$$

et, de façon cohérente avec les notations précédentes, nous posons $U^\star_{[i,j]|\ell} = \sum_{k=i}^{j} u^\star_{k|\ell}$ et $U^\star_{[>i]|\ell} = \sum_{k>i} u^\star_{k|\ell}$.

PROPOSITION 7.5. (EXPRESSION DES SÉRIES DE DIRICHLET POUR LES TRIES HYBRIDES, CAS MARKOVIEN). *Prenant la notation (7.16), pour une source* (S, F) *avec un codage* σ *monotone, les séries de Dirichlet s'expriment sous la forme*

$$\Lambda^{\langle B\rangle}(F,s) = 2\sum_{\substack{\ell,i,j\\ i<j}} u^{\star}_{i|\ell}u^{\star}_{j|\ell}{U^{\star}_{[i,j]|\ell}}^{s-2}\left[I + {}^t\mathbf{e}_\ell\left(I-\mathfrak{G}_s^{\langle B\rangle}\right)^{-1}\mathfrak{M}_s\right] \tag{7.21}$$
$$[L_s^{\langle B\rangle}](h^-_{i|\ell}, h^+_{i|\ell}, h^-_{j|\ell}, h^+_{j|\ell}),$$

$$\Lambda^{\langle L\rangle}(F,s) = \sum_{\ell,i} U^{\star}_{[>i]|\ell}u^{\star}_{i|\ell}\ \left[I + {}^t\mathbf{e}_\ell\left(I-\mathfrak{G}_s^{\langle L\rangle}\right)^{-1}\mathfrak{M}_s\right] \tag{7.22}$$
$$[L_s^{\langle L\rangle}](h^-_{i|\ell}, h^+_{i|\ell}, h^+_{r|\ell}).$$

En conclusion, comme dans le cas de base, les séries de Dirichlet concernées pour les tries font intervenir trois opérateurs de Ruelle généralisés (qui sont maintenant matriciels) $\mathfrak{G}_s^{\langle A\rangle}$, $\mathfrak{G}_s^{\langle L\rangle}$, $\mathfrak{G}_s^{\langle B\rangle}$, plus l'opérateur (également matriciel) correspondant au système dynamique initial $\mathfrak{M}_s$.

L'étude de ces séries de Dirichlet (position et nature des pôles) est étroitement liée aux propriétés spectrales des opérateurs de Ruelle (généralisés) qui les engendrent. Pour l'étude de la hauteur, le comportement asymptotique de $\Lambda_k^{\langle A\rangle}$ (lorsque k tend vers l'infini) est relié aux propriétés spectrales dominantes de l'opérateur de Ruelle $\mathfrak{G}_s^{\langle A\rangle}$. La taille et la longueur de cheminement fnt intervenir des quasi-inverses $(I-\mathfrak{G}_s)^{-1}$ de différents opérateurs de Ruelle $\mathfrak{G}_s$. L'analyse asymptotique des différents paramètres nécessite une localisation précise des pôles de ces séries de Dirichlet. De tels pôles proviennent de valeurs de s pour lesquelles $(I-\mathfrak{G}_s)^{-1}$ est singulière, c'est à dire des valeurs pour lesquelles 1 est une valeur propre de $\mathfrak{G}_s$. Ainsi, les pôles de la série de Dirichlet sont aussi reliés aux propriétés spectrales des opérateurs de transfert.

7.2 Opérateurs générateurs : propriétés d'analyse fonctionnelle

Les opérateurs généralisés multisécants appartiennent à la catégorie des opérateurs nucléaires (définis dans la Section 7.2.1) et ont, en particulier, un spectre discret. De tels opérateurs, agissant pourtant sur des espaces de dimension infinie, se comportent en bien des aspects comme des matrices finies. Ainsi on peut définir la trace mais aussi le déterminant de Fredholm qui généralise le polynôme caractéristique. Tout d'abord, nous introduisons les opérateurs de composition qui sont les «briques» de base pour construire des opérateurs de transfert et ont un spectre précisément connu (Section 7.2.2). Le spectre des opérateurs de transfert est alors déterminé grâce au spectre de ces opérateurs de composition au moyen de formules de traces (Section 7.2.2). Les opérateurs généralisés possèdent également de fortes propriétés de positivité de type Perron-Frobenius pour des valeurs réelles du paramètre s. Cette positivité entraîne l'existence de propriétés spectrales dominantes (Section 7.2.3) qui peuvent être reliées à l'entropie et à la probabilité de coïncidence de la source (Section 7.2.3). La «propriété de quasi-puissance» donne des informations précieuses sur les itérés de ces opérateurs (Section 7.2.3). D'autres propriétés peuvent être transférées aux séries de Dirichlet des intervalles fondamentaux survenant dans le cadre de l'analyse des tries (Sections 7.2.4, 7.2.3, 7.2.5).

7.2.1 Nucléarité, formule de trace et déterminant de Fredholm

La notion de nucléarité telle qu'elle a été introduite par Grothendieck [50, 51] est la suivante : Soit B un espace de Banach et $B^\star$ son espace dual. Un opérateur $\mathcal{M} : B \to B$ est nucléaire d'ordre 0 s'il admet la représentation

$$\mathcal{M}[f] = \sum_{i\in I} \mu_i \; e_i^\star(f) \; e_i \;\; \text{pour tout } f \in B,$$

avec $e_i \in B$, $e_i^\star \in B^\star$ tel que $||e_i|| = ||e_i^\star|| = 1$ et les μ_i p-sommables pour tout $p > 0$ (i.e., $\sum |\mu_i|^p < +\infty$). L'algèbre matricielle peut en grande partie s'étendre à de tels opérateurs. En particulier, la trace est définie par

$$\mathrm{Tr}\; \mathcal{L} = \sum_{i\in I} \mu_i \; e_i^\star(e_i), \quad \text{ou encore} \quad \mathrm{Tr}\; \mathcal{L} = \sum_{i\in I} \lambda_i,$$

où les λ_i sont les valeurs propres de $\mathcal{L}$, comptées avec leurs multiplicités algébriques. Les traces des itérés de $\mathcal{L}$ sont également bien définies, ainsi que l'analogue du polynôme caractéristique appelé déterminant de Fredholm,

$$\mathrm{Tr}\; \mathcal{L}^k = \sum_{i\in I} \lambda_i^k, \qquad F(\mathcal{L}, u) := \det(I - u\mathcal{L}) := \prod_{i\in I} (1 - \lambda_i u), \tag{7.23}$$

où les λ_i sont les valeurs propres de $\mathcal{L}$ (en prenant en compte les multiplicités). Il existe une relation importante entre le déterminant de Fredholm et les traces des itérés,

$$\det(I - u\mathcal{L}) = \exp\Big[\mathrm{Tr} \log(I - u\mathcal{L})\Big] = \exp\Big[-\sum_{k=1}^{\infty} \frac{u^k}{k} \mathrm{Tr}\; \mathcal{L}^k \Big] \tag{7.24}$$

En effet, le logarithme de $(I - \mathcal{L})$ pour $\mathcal{L}$ de norme inférieure à 1 est défini par $\log(I - \mathcal{L}) = \sum_{k\geq 0} (-1)^k \mathcal{L}^k / k$. Ainsi, on obtient

$$\begin{aligned} \mathrm{Tr} \log(I - u\mathcal{L}) &= \mathrm{Tr}(-\sum_{k=1}^{\infty} \frac{u^k}{k} \mathcal{L}^k) = -\sum_{k=1}^{\infty} \frac{u^k}{k} \mathrm{Tr}(\mathcal{L}^k) \\ &= \sum_{k=1}^{\infty} \frac{u^k}{k} \sum_{i\in I} \lambda_i^k = \sum_{i\in I} \log(1 - u\lambda_i) = \log\det(I - u\mathcal{L}). \end{aligned}$$

Cette formule étend la formule de Jacobi en algèbre matricielle : $\log \circ \det = \mathrm{Tr} \circ \log$. Ces propriétés donnent accès aux propriétés spectrales des opérateurs nucléaires d'ordre 0.

7.2.2 Spectre des opérateurs

Opérateurs de composition

On considère ici les opérateurs $\mathcal{G}_{s,h}$ relatifs à une branche inverse (cela recouvre le cas de base avec h_i, le cas markovien $h_{j|i}$ mais c'est également valable pour toute branche $h_{\boldsymbol{w}}$). Chaque opérateur composante $\mathcal{G}_{s,h}$ défini par

$$\mathcal{G}_{s,h}[f] := \widetilde{h}^s \; f \circ h$$

est un opérateur dit de composition. Du fait que chaque branche h satisfait les propriétés de contraction (d_1) et (d_2) de la définition 4.1, il existe un voisinage $\mathcal{V}$ de $\overline{\mathcal{I}} = [0, 1]$ tel que h

et $|h'|$ se prolongent analytiquement sur $\mathcal{V}$; h envoie la fermeture $\overline{\mathcal{V}}$ de $\mathcal{V}$ dans $\mathcal{V}$ strictement ; il existe $\delta < 1$ pour lequel $0 < |\widetilde{h}(z)| \leq \delta$ pour tout $z \in \mathcal{V}$.

Dès lors, l'opérateur $\mathcal{G}_{s,h}$ agit sur l'espace $A_\infty(\mathcal{V})$ des fonctions f qui sont holomorphes dans le domaine $\mathcal{V}$ et sont continues sur la fermeture $\overline{\mathcal{V}}$. Muni de la norme sup

$$||f|| = \sup\{|f(u)|;\ u \in \overline{\mathcal{V}}\},$$

$A_\infty(\mathcal{V})$ est un espace de Banach. De tels opérateurs ont été étudiés intensivement par plusieurs auteurs (Schwartz [88], Shapiro et Taylor [92], Shapiro [91]).

Chaque branche h satisfaisant les conditions (d_1) et (d_2) de la définition 4.1, l'opérateur généralisé $\mathfrak{G}_{s,h}$ agit sur l'espace $\mathfrak{A}_\infty(\mathcal{V})$ des fonctions F qui sont holomorphes sur le domaine $\mathcal{V}^m$ et sont continues sur la fermeture $\overline{\mathcal{V}}^m$. Muni de la norme-sup

$$||F|| = \sup\{|F(u_1, u_2, \ldots, u_m)|;\ (u_1, u_2, \ldots, u_m) \in \overline{\mathcal{V}}^m\},$$

$\mathfrak{A}_\infty(\mathcal{V})$ est un espace de Banach.

Tous les opérateurs de composition (étendus grâce à l'utilisation de multisécantes) $\mathfrak{G}_{s,h}$ relatifs à la même branche inverse h ont les mêmes valeurs propres. Cependant le degré m de l'extension considérée (le nombre de variables considéré des fonctions sur lesquelles on va appliquer l'opérateur) détermine les multiplicités de ces valeurs propres.

PROPOSITION 7.6. *Soit $\alpha(h) = \widetilde{h}(\underline{h})$ la valeur de $\widetilde{h}$ au point fixe $\underline{h}$ de h et $\epsilon(h)$ le signe de la dérivée h' sur $\mathcal{I}$. Le spectre de l'opérateur $\mathfrak{G}_{s,h}$, extension de degré m, est alors formé des valeurs propres $\mu_\ell, (\ell \geq 0)$,*

$$\mu_\ell := \alpha(h)^s [\epsilon(h)\alpha(h)]^\ell. \tag{7.25}$$

Chaque valeur propre μ_ℓ apparaît dans le spectre $\operatorname{Sp} \mathfrak{G}_{s,h}$ avec une multiplicité $\binom{\ell+m-1}{m-1}$. Par conséquent, la formule de trace pour $\mathfrak{G}_{s,h}$ donne

$$\operatorname{Tr} \mathfrak{G}_{s,h} = \frac{\alpha(h)^s}{(1 - \epsilon(h)\alpha(h))^m}.$$

Preuve. La preuve utilise un théorème de Mayer [74] et le généralise. Ici, la notation $\mathsf{Multi}\,[A]$ désigne l'ensemble des multiensembles construit sur le multiensemble A. Plus précisément, pour un multiensemble $A = \{a_1, a_2, a_3, \ldots, a_r\}$, (où l'égalité entre les symboles a_i est possible), on a avec des notations empruntées aux langages réguliers et déjà introduites au chapitre 6

$$\mathsf{Multi}\,[A] = \prod_{1 \leq i \leq r} a_i^*.$$

THÉORÈME (MAYER). *Soit Ω un ouvert de $\mathbb{C}^m$, et $B_\infty(\Omega)$ l'ensemble des fonctions qui sont holomorphes sur Ω et continues sur $\overline{\Omega}$. Soit $\mathcal{R}_\phi$ l'opérateur défini sur $\mathcal{B}_\infty(\Omega)$ par $\mathcal{R}_\phi[f] := \phi\ f \circ \psi$, où $\phi \in B_\infty(\Omega)$ et ψ envoie strictement Ω dans lui-même. Le spectre de $\mathcal{R}_\phi$ est*

$$\operatorname{Sp}(\mathcal{R}_\phi) = \phi(\underline{z})\ .\ \mathsf{Multi}\ [\operatorname{Sp}(\psi'(\underline{z})]$$

où $\underline{z}$ est l'unique point fixe de ψ dans Ω et ψ' est la différentielle de ψ.

Les opérateur composantes $\mathfrak{G}_{s,h}$ remplissent bien les conditions du théorème : leurs spectres sont précisément déterminés. Il est à noter que dans le cadre de l'analyse des tries, seuls les extensions de degré $m = 1, 2, 3, 4$ nous intéressent.

Dans le cas $m = 1$ où l'opérateur $\mathfrak{G}_{s,h}$ coïncide exactement avec l'opérateur de Ruelle usuel $\mathcal{G}_{s,h}$, le théorème précédent s'applique avec $\phi = \widetilde{h}^s$, $\psi = h$, et $\mathrm{Sp}(\psi'(\underline{z})) = \{h'(\underline{h})\} = \{\epsilon(h)\alpha(h)\}$, si bien que

$$\mathrm{Sp}\, \mathcal{G}_{s,h} = \{\alpha(h)^s(\epsilon(h)\alpha(h))^\ell \mid \ell \in \mathbb{N}\}$$

est exactement l'ensemble des μ_ℓ définis dans (7.25).

Dans le cas général où l'opérateur $\mathfrak{G}_{s,h}$ agit sur un espace à m variables (extension de degré m), le théorème de Mayer s'applique avec

$$\phi = \widetilde{H}_s[h], \qquad \psi = V_m[h].$$

La différentielle $\psi'(x_1, \ldots, x_m)$ est alors la matrice diagonale avec des coefficients $h'(x_1), h'(x_2), \ldots, h'(x_m)$. Le point fixe de $V_m[h]$ est le point$(\underline{h}, \underline{h}, \ldots, \underline{h})$, si bien que

$$\mathrm{Sp}\, V'_m[h]\ (\underline{h}, \underline{h}, \ldots, \underline{h}) = \{h'(\underline{h})\}^{[m]},$$

où la notation $A^{[m]}$ désigne le multiensemble obtenu en répétant m fois chaque élément du (multi)ensemble A. La multiplicité de la valeur propre μ_ℓ dans le spectre de $\mathfrak{G}_{s,h}$ est alors égale au nombre de mots de longueur ℓ dans le langage $a_1^* a_2^* \cdots a_m^*$, c.-à-d.

$$[z^\ell]\ \frac{1}{(1-z)^m} = \binom{\ell + m - 1}{m - 1}.$$

En conséquence, on obtient des formules de trace faisant intervenir la quantité $\alpha(h) = \widetilde{h}(\underline{h})$

$$\mathrm{Tr}\, \mathfrak{G}_{s,h} = \frac{\alpha(h)^s}{(1 - \epsilon(h)\alpha(h))^m}.$$

□

Ainsi, les spectres de $\mathcal{G}_{s,h}$ et $\mathfrak{G}_{s,h}$ contiennent les mêmes éléments. Nous comparons maintenant ces spectres de façon plus précise, grâce aux formules de trace. Avec la notation $p_m(\ell) = \binom{\ell+m-2}{m-2}$, la trace de l'opérateur généralisé peut être exprimée à l'aide de la trace de l'opérateur classique,

$$\mathrm{Tr}\, \mathfrak{G}_{s,h} = \frac{\mathrm{Tr}\, \mathcal{G}_{s,h}}{(1 - \epsilon(h)\alpha(h))^{m-1}} = \sum_{\ell \geq 0} p_m(\ell)\ (\epsilon(h)\alpha(h))^\ell\, \mathrm{Tr}\, \mathcal{G}_{s,h}.$$

Nous introduisons donc la version signée de l'opérateur de Ruelle classique $\widetilde{\mathcal{G}}_s$ dont les opérateurs composantes sont définis en introduisant le signe $\epsilon(h)$ de la dérivée h',

$$\widetilde{\mathcal{G}}_{s,[i]} := \epsilon(h_i)\ \mathcal{G}_{s,[i]}. \tag{7.26}$$

La propriété de multiplicativité de ϵ pour les branches inverses

$$\epsilon(h \circ g) = \epsilon(h)\epsilon(g)$$

entraîne alors

$$\widetilde{\mathcal{G}}_s^k = \sum_{\boldsymbol{w} \in \mathcal{M}^k} \epsilon(h_{\boldsymbol{w}})\ \mathcal{G}_{s,h_{\boldsymbol{w}}} = \sum_{\boldsymbol{w} \in \mathcal{M}^k} \widetilde{\mathcal{G}}_{s,h_{\boldsymbol{w}}}. \tag{7.27}$$

La trace de l'opérateur généralisé $\mathfrak{G}_{s,h}$ dans une extension de degré m, est exprimée à l'aide des traces des opérateurs $\mathcal{G}_{s+\ell,h}$, $\widetilde{\mathcal{G}}_{s+\ell,h}$,

$$\operatorname{Tr} \mathfrak{G}_{s,h} = \sum_{\substack{\ell \text{ pair} \\ \ell \geq 0}} p_m(\ell) \operatorname{Tr} \mathcal{G}_{s+\ell,h} + \sum_{\substack{\ell \text{ impair} \\ \ell \geq 0}} p_m(\ell) \operatorname{Tr} \widetilde{\mathcal{G}}_{s+\ell,h}. \tag{7.28}$$

Opérateurs de transfert

À ce stade, l'espace fonctionnel sur lequel agit l'opérateur $\mathfrak{G}_s$ doit être précisé. L'opérateur est ici déterminé par un entier m (le degré de l'extension) et une fonction muliséCante $\widetilde{H}_s$ (ou plutôt son prolongement analytique). Dans le cas markovien, nous nous limiterons au cas des alphabets finis, mais autoriserons un alphabet qui soit infini dénombrable dans le cas de base. Dans cette situation, la possibilité de choisir le même ensemble ouvert $\mathcal{V}$ pour toutes les branches h en conjonction avec la condition de convergence (d_3) entraîne de «bonnes» propriétés pour l'opérateur de Ruelle $\mathfrak{G}_s$ où s appartient au demi-plan de convergence $\operatorname{Re}(s) > \gamma$. Nous noterons dans la suite $\mathcal{V}_{\mathbb{R}}$ l'intersection de $\mathcal{V}$ avec l'axe réel. La multisécante $\widetilde{H}_s[h]$ définie dans (7.8) a une partie réelle strictement positive sur $\mathcal{V}^m$ (car elle est positive sur $\mathcal{I}^m$ et ne s'annule pas sur $\mathcal{V}^m$), et l'opérateur $\mathfrak{G}_s$ est bien défini pour tout nombre complexe s dans le demi-plan $\operatorname{Re}(s) > \gamma$.

Dans le cas de base, les opérateurs $\mathfrak{G}_s$ agissent sur l'espace $\mathfrak{A}_\infty(\mathcal{V})$ défini comme l'ensemble des fonctions qui sont holomorphes sur $\mathcal{V}^m$ et sont continues sur la fermeture $\overline{\mathcal{V}}^m$, muni de la norme-sup. Dans le cas markovien, les opérateurs $\mathfrak{G}_s$ agissent sur l'espace $\mathfrak{A}_\infty(\mathcal{V})^r$. Ces deux espaces fonctionnels sont des espaces de Banach. Les opérateurs composantes $\mathfrak{G}_{s,h}$ sont nucléaires d'ordre 0, donc les opérateurs $\mathfrak{G}_s$ sont nucléaires d'ordre 0. En particulier ils sont bornés, compacts et leurs spectres sont discrets (sauf au point d'accumulation en 0).

La proposition suivante met en relation le spectre des opérateurs généralisés avec les spectres des opérateurs classiques (signés et non signés).

PROPOSITION 7.7. *Le spectre de l'opérateur de Ruelle généralisé $\mathfrak{G}_s$ est en relation avec celui des opérateurs classiques $\mathcal{G}_s$, $\widetilde{\mathcal{G}}_s$*

$$\operatorname{Sp} \mathfrak{G}_s = \left(\bigcup_{\substack{\ell \text{ pair} \\ \ell \geq 0}} (\operatorname{Sp} \mathcal{G}_{s+\ell})^{[p_m(\ell)]} \right) \cup \left(\bigcup_{\substack{\ell \text{ impair} \\ \ell \geq 0}} \left(\operatorname{Sp} \widetilde{\mathcal{G}}_{s+\ell} \right)^{[p_m(\ell)]} \right),$$

où $p_m(\ell) = \binom{l+m+2}{m-2}$. Ici, la réunion est prise au sens de la réunion des multiensembles et la notation $A^{[p]}$ désigne le multiensemble obtenu en répétant p fois chaque élément du (multi)ensemble A.

Preuve. Dans le cas markovien, la trace du k^{e} itéré de $\mathcal{G}_s$ (resp. $\mathfrak{G}_s$) est la somme des traces des éléments diagonaux de la matrice $\mathcal{G}_s^k$ (resp. $\mathfrak{G}_s^k$) ; Ces éléments diagonaux mettent en jeu uniquement des branches inverses $h_{\boldsymbol{w}}$ qui sont cycliques : ce sont les branches inverses pour lesquelles les préfixes associés commencent et finissent par le même symbole. Finalement, dans les deux cas, les formules de trace font intervenir l'ensemble $\mathcal{C}_k$ (défini dans la section 4.4) qui est l'ensemble des préfixes de longueur k (dans le cas de base) ou l'ensemble des préfixes de longueur k qui commencent et finissent par la même lettre (dans le cas markovien),

$$\operatorname{Tr} \mathcal{G}_s^k = \sum_{\boldsymbol{w} \in \mathcal{C}_k} \operatorname{Tr} \mathcal{G}_{s,[\boldsymbol{w}]}, \qquad \operatorname{Tr} \mathfrak{G}_s^k = \sum_{\boldsymbol{w} \in \mathcal{C}_k} \operatorname{Tr} \mathfrak{G}_{s,[\boldsymbol{w}]}.$$

Dès lors, la relation (7.28) s'étend aux puissances (ou itérés) des opérateurs de transfert,

$$\operatorname{Tr} \mathfrak{G}_s^k = \sum_{\substack{\ell \text{ pair} \\ \ell \geq 0}} p_m(\ell) \operatorname{Tr} \mathcal{G}_{s+\ell}^k + \sum_{\substack{\ell \text{ impair} \\ \ell \geq 0}} p_m(\ell) \operatorname{Tr} \widetilde{\mathcal{G}}_{s+\ell}^k.$$

Grâce à la formule de la trace (7.24), le déterminant de Fredholm $\mathfrak{F}(s,u) := \det(I - u\mathfrak{G}_s)$ peut être exprimé en termes de traces d'itérés des opérateurs

$$\log\det(I - u\mathfrak{G}_s) = \sum_{\substack{\ell \text{ pair} \\ \ell \geq 0}} p_m(\ell) \log\det(I - u\mathcal{G}_{s+\ell}) + \sum_{\substack{\ell \text{ impair} \\ \ell \geq 0}} p_m(\ell) \log\det(I - u\widetilde{\mathcal{G}}_{s+\ell}).$$

Le déterminant de Fredholm de l'opérateur de Ruelle généralisé $\mathfrak{G}_s$ agissant sur un espace fonctionnel à m variables satisfait alors

$$\det(I - u\,\mathfrak{G}_s) = \prod_{\substack{\ell \text{ pair} \\ \ell \geq 0}} [\det(I - u\,\mathcal{G}_{s+\ell})]^{p_m(\ell)} \prod_{\substack{\ell \text{ impair} \\ \ell \geq 0}} \Big[\det(I - u\,\widetilde{\mathcal{G}}_{s+p(\ell)})\Big]^{p_m(\ell)},$$

ce qui donne finalement la relation entre le spectre de $\mathfrak{G}_s$ et les spectres de $\mathcal{G}_s$ et $\widetilde{\mathcal{G}}_s$. □

7.2.3 Positivité

Propriétés spectrales dominantes pour s réel (Perron-Frobenius)

Lorsque $s = \sigma > \gamma$ est réel, les opérateurs $\mathcal{G}_s, \mathfrak{G}_s$ satisfont de fortes propriétés de positivité de type Perron-Frobenius [65].

PROPOSITION 7.8. *Pour $s > \gamma$ réel, chaque opérateur $\mathfrak{G}_s$ possède une unique valeur propre dominante (i.e., une valeur propre de plus grand module) $\lambda(s)$ positive et de multiplicité égale à 1.*

Remarque. En fait on peut montrer plus. Dans le cadre de l'analyse des tries, on aura besoin de savoir que toutes les extensions considérées ont la même valeur propre dominante. Cela fera l'objet de la proposition suivante 7.13.

Preuve. Soit B un espace de Banach, réel ou complexe, et $K \subset B$. L'ensemble K est un cône propre si $\rho\,K \subset K$ pour $\rho > 0$ et $K \cap -K = \{0\}$. Un cône propre est dit reproductif si $B = K - K$, i.e., tout élément g de B est la différence de deux éléments de K. Le cône K définit un ordre partiel $\leqq$ dans B : $x \leqq y$ ssi $y - x \in K$.
Un opérateur linéaire $\mathcal{L} : B \to B$ est positif par rapport à K ssi $\mathcal{L}K \subset K$, ce qui signifie que T laisse le cône invariant. Soit $u_0 \in K$, $u_0 \neq 0$. Un opérateur positif $\mathcal{L} : B \to B$ est appelé u_0-positif, pour u_0 dans l'intérieur $\overset{\circ}{K}$ de K, s'il existe pour tout $f \in K \setminus \{0\}$, deux entiers p, q et deux réels $\alpha, \beta > 0$ pour lesquels on a

$$\alpha\, u_0 \leqq \mathcal{L}^p[f] \text{ et } \mathcal{L}^q[f] \leqq \beta\, u_0, \tag{7.29}$$

où l'ordre est défini par rapport à K : $f \leqq g \Leftrightarrow g - f \in K$. Ces notions étant définies, on peut énoncer le théorème de Krasnolesjii dans une formulation de Mayer dans [5].

THÉOREME DE KRASNOELSKII (POSITIVITÉ). *Soit $\mathcal{L} : B \to B$ un opérateur compact u_0-positif. Alors $\mathcal{L}$ satisfait une propriété de type Perron-Frobenius : il a un unique vecteur propre dans $\overset{\circ}{K}$ et la valeur propre correspondante est simple, positive, et de module strictement supérieur au module des autres valeurs propres de $\mathcal{L}$.*

Nous appliquons tout d'abord le résultat de Krasnoselsky pour l'opérateur $\mathcal{G}_s$ dans le cas de base. Nous commençons par préciser les espaces fonctionnels considérés. Nous avons déjà défini l'espace de Banach $A_\infty(\mathcal{V})$ des fonctions holomorphes sur $\mathcal{V}$ et continues sur $\overline{\mathcal{V}}$. Nous définissons l'intervalle $\mathcal{V}_\mathbb{R} = \mathcal{V} \cap \mathbb{R}$ et le sous-espace réel $A_\infty(\mathcal{V}_\mathbb{R})$ des fonctions de $A_\infty(\mathcal{V})$ à valeurs réelles sur l'intervalle $\mathcal{V}_\mathbb{R}$)

$$A_\infty(\mathcal{V}_\mathbb{R}) = \{f \in A_\infty(\mathcal{V}) \mid f(z) \in \mathbb{R} \text{ pour tout } z \in R\}\,.$$

Clairement, $A_\infty(\mathcal{V}_\mathbb{R})$ est un espace de Banach réel avec la norme sup induite. L'ensemble $A^+_\infty(\mathcal{V}_\mathbb{R}) \subset A_\infty(\mathcal{V})$ des fonctions positives sur l'intervalle $\mathcal{V}_\mathbb{R}$

$$A^+_\infty(\mathcal{V}_\mathbb{R}) = \{f \in A_\infty(\mathcal{V}_\mathbb{R}) \mid f(z) \geq 0 \text{ pour tout } z \in R\}\,.$$

Pour s réel, $\mathcal{G}_s$ agit sur $A^+_\infty(\mathcal{V}_\mathbb{R})$ qui est un cône propre est reproductif. Si f est une fonction positive sur $\mathcal{V}_\mathbb{R}$, ρf avec $\rho > 0$ l'est également : le cône est propre. Toute fonction $f \in A^+_\infty(\mathcal{V}_\mathbb{R})$ peut s'écrire $2f - f$, ce qui prouve que le cône est reproductif. L'intérieur du cône $\overset{\circ}{A}_+$, est formé des éléments $f \in A^+_\infty(\mathcal{V}_\mathbb{R})$ qui sont strictement positifs sur le segment de l'axe des réels $\mathcal{V}_\mathbb{R}$.

La démonstration de la proposition nécessite plusieurs étapes que nous formulons à l'aide de lemmes. Le premier lemme concerne l'opérateur $\mathcal{G}_{s\mathbb{R}}$ la restriction de $\mathcal{G}_s$ au sous-espace $A_\infty(\mathcal{V}_\mathbb{R})$.

LEMME 7.9. *L'opérateur $\mathcal{G}_s$ définit un opérateur $\mathcal{G}_{s\mathbb{R}}$ nucléaire d'ordre 0 dans l'espace de Banach réel $A_\infty(\mathcal{V}_\mathbb{R})$ dont la trace est*

$$\operatorname{Tr} \mathcal{G}_{s\mathbb{R}} = \operatorname{Tr} \mathcal{G}_s.$$

Preuve. En effet, nous savons que $\mathcal{G}_s$ est un opérateur nucléaire d'ordre 0 ([74]) et admet la représentation

$$\mathcal{G}_s[f] = \sum_{i \in I} \mu_i \; e^\star_i(f) \; e_i \;\; \text{pour tout } f \in A_\infty(\mathcal{V}),$$

avec $e_i \in A_\infty(\mathcal{V})$, $e^\star_i \in A_\infty(\mathcal{V})^\star$ tel que $||e_i|| = ||e^\star_i|| = 1$ et les μ_i p-sommables pour tout $p > 0$ (i.e., $\sum |\mu_i|^p < +\infty$). La définition de l'opérateur $\mathcal{G}_s$ à l'aide des branches inverses contractantes à valeurs réelles permet d'établir que les μ_i peuvent être choisis réels, les e_i dans $A_\infty(\mathcal{V}_\mathbb{R})$ et les $e^\star_i$ à valeurs réelles sur $A_\infty(\mathcal{V}_\mathbb{R})$. Ainsi les deux opérateurs $\mathcal{G}_s$ et $\mathcal{G}_{s\mathbb{R}}$ ont exactement la même représentation. Cela montre que la trace de $\mathcal{G}_{s\mathbb{R}}$ a la même expression que celle de $\mathcal{G}_s$. □

Nous voulons montrer que les opérateurs $\mathcal{G}_s$ et $\mathcal{G}_{s\mathbb{R}}$ ont le même spectre. Le spectre de l'opérateur $\mathcal{G}_{s\mathbb{R}}$ dans l'espace $A_\infty(\mathcal{V}_\mathbb{R})$ est par définition le spectre de l'opérateur $\mathcal{G}_{s\mathbb{R}}$ dans le sous-espace $A_\infty(\mathcal{V}_\mathbb{R}) + iA_\infty(\mathcal{V}_\mathbb{R})$ de $A_\infty(\mathcal{V})$. Par ailleurs, $\mathcal{G}_{s\mathbb{R}}$ est également un opérateur nucléaire d'ordre 0 dans cet espace complexe qui est trivialement un sous-espace de l'espace de Banach $A_\infty(\mathcal{V})$.

Ainsi chaque valeur propre de l'opérateur $\mathcal{G}_{s\mathbb{R}}$ est également une valeur propre de l'opérateur $\mathcal{G}_s$ dans $A_\infty(\mathcal{V}$.

Désignons par Υ l'ensemble des valeurs propres de $\mathcal{G}_s$ différentes de 0 qui n'appartiennent pas au spectre $\operatorname{Sp} \mathcal{G}_{s\mathbb{R}}$. Les deux opérateurs étant nucléaires d'ordre 0, on a

$$\operatorname{Tr} \mathcal{G}^n_{s\mathbb{R}} = \sum_{\lambda \in \operatorname{Sp} \mathcal{G}_{s\mathbb{R}}} \lambda^n, \text{ et } \operatorname{Tr} \mathcal{G}^n_s = \sum_{\lambda \in \operatorname{Sp} \mathcal{G}_{s\mathbb{R}}} \lambda^n + \sum_{\lambda \in \Upsilon} \lambda^n.$$

Grâce au lemme 7.9, nous avons alors que pour tout $n \geq 1$,

$$\sum_{\lambda \in \Upsilon} \lambda^n = 0.$$

On peut alors appliquer le lemme issu de [73]

LEMME 7.10. *Soit Υ un ensemble de nombres complexes tels que $\sum_{\lambda\in\Upsilon} |\lambda| < \infty$ et $\sum_{\lambda\in\Upsilon} \lambda^n = 0$ pour tout $n \geq 1$. Alors $\lambda = 0$ pour tout $\lambda \in \Upsilon$.*

Preuve. Le fait que $\sum_{\lambda\in\Upsilon} |\lambda| < \infty$ implique que, pour z suffisamment petit, la fonction g suivante est holomorphe

$$g(z) = \exp \sum_{i=1}^{\infty} \frac{1}{n} \sum_{\lambda\in\Upsilon} \lambda^n z^n.$$

Puisque $\sum_{\lambda\in\Upsilon} \lambda^n = 0$ la fonction est donc $g \equiv 1$. On calcule le terme droit de l'égalité pour z suffisamment petit

$$exp \sum_{i=1}^{\infty} \frac{1}{n} \sum_{\lambda\in\Upsilon} \lambda^n z^n = \prod_{\lambda\in\Upsilon} (1 - \lambda z)^{-1}.$$

Cette fonction doit être une série entière sur tout le plan complexe, puisque le terme gauche de l'égalité l'est. Cela est possible seulement si $\lambda = 0$ pour tout $\lambda \in \Upsilon$. □

Ceci montre le lemme suivant.

LEMME 7.11. *Les spectres des opérateurs $\mathcal{G}_s$ et $\mathcal{G}_{s\mathbb{R}}$ sont les mêmes.*

Les spectres des deux opérateurs étant identiques, il suffit de caractériser le spectre de $\mathcal{G}_{s\mathbb{R}}$.

LEMME 7.12. *L'opérateur $\mathcal{G}_{s\mathbb{R}} : A_\infty(\mathcal{V}_\mathbb{R}) \to A_\infty(\mathcal{V}_\mathbb{R})$ est u_0-positif par rapport au cône $K = A_\infty(\mathcal{V}_\mathbb{R})$.*

Preuve. Les branches inverses étant contractantes, il est clair que $\mathcal{G}_{s\mathbb{R}}$ laisse le cône invariant et donc que l'opérateur est positif. Désignons par u_0 la fonction constante égale à 1. Alors clairement $u_0 \in K$ et $u_0 \neq 0$. Soit $f \in K$ ($f \neq 0$). En posant $\alpha_f = \sup_{z\in\mathcal{V}_\mathbb{R}} \|G_s[f](z)|$, on a

$$\mathcal{G}_s[f] \leq \alpha_f u_0. \tag{7.30}$$

On a donc montré la borne supérieure de (7.29).

Pour la borne inférieure, soit $f \in K \subset \{0\}$ et supposons que, pour tout entier p il existe $x_p \in \mathcal{V}_\mathbb{R}$ pour lequel $\mathcal{G}^p_{s\mathbb{R}}[f](x_p) = 0$. Alors, f vaut 0 en chaque point $h(x_p)$ associé à une branche inverse h de hauteur p. Dans le cas de l'alphabet infini, f s'annule donc en une infinité de points distincts. Étant analytique, la fonction f est donc identiquement nulle. Dans le cas d'un alphabet fini, le nombre de points où f s'annule est également non borné (il suffit de faire croître p). Là encore, la fonction f est donc nulle. On aboutit à une contradiction, il existe donc p tel que pour tout $x \in \mathcal{V}_R$, $\mathcal{G}^p_{s\mathbb{R}}[f](x)$ ne s'annule pas. Posons $\beta_f = \inf_{x\in\mathcal{V}_\mathbb{R}} \mathcal{G}^p_{s\mathbb{R}}[f](x)$. Ainsi nous obtenons

$$\mathcal{G}^p_{s\mathbb{R}}[f] \geqq \beta_f u_0, \quad \text{avec } \beta_f > 0. \tag{7.31}$$

Les équations (7.30) et (7.31) entraîne le fait que $\mathcal{G}_{s\mathbb{R}}$ est u_0-positif (voir [65] pour plus de détails sur les opérateurs u_0-positifs). □

Le théorème de Krasnoselsky s'applique donc à $\mathcal{G}_{s\mathbb{R}}$. L'opérateur $\mathcal{G}_{s\mathbb{R}}$ (et donc $\mathcal{G}_s$) admet donc une unique valeur propre dominante $\lambda(s)$ strictement positive. On peut choisir le vecteur propre dominant ψ_s dans le cône $\overset{\circ}{K}$, c.-à-d. que ψ_s est strictement positive sur $\mathcal{V}_{\mathbb{R}}$. Ce vecteur propre est également propre pour $\mathcal{G}_s$.

Cette preuve se généralise au cas markovien en considérant l'espace $A_\infty(\mathcal{V})^r$ et aussi au cas des opérateurs généralisés en considérant les espaces $\mathfrak{A}_\infty(\mathcal{V})$ et $\mathfrak{A}_\infty(\mathcal{V})^r$. □

A priori, cette valeur propre dominante est dépendante de l'extension choisie. Nous prouvons dans cette section que toutes les extensions de $\mathcal{G}_s$ partagent la même valeur propre dominante.

L'opérateur $\mathfrak{G}_s$ étant compact, son spectre est discret et il y a un «saut spectral» entre la valeur propre dominante et le reste du spectre. Cela permet de décomposer $\mathfrak{G}_s$ sous la forme

$$\mathfrak{G}_s = \lambda(s)\mathfrak{P}_s + \mathfrak{N}_s.$$

Ici, $\mathfrak{P}_s$ est la projection sur le sous-espace propre dominant, et $\mathfrak{N}_s$ est un opérateur relatif au reste du spectre. L'opérateur $\mathfrak{N}_s$ a donc un rayon spectral strictement inférieur à la valeur propre dominante.

De façon plus générale, $\mathfrak{G}_s^k$ se décompose comme

$$\mathfrak{G}_s^k = \lambda(s)^k\, \mathfrak{P}_s + \mathfrak{N}_s^k. \tag{7.32}$$

Cette relation est vraie en particulier pour l'opérateur de Ruelle classique $\mathcal{G}_s$ dont la valeur dominante est notée $\lambda_1(s)$. Les relations précédentes, ainsi que les propriétés de positivité de la projection sur le sous-espace propre dominant, entraînent les égalités

$$\lambda(s) = \lim_{k\to\infty} \mathfrak{G}_s^k\,[1](0,\dots,0) \qquad \lambda_1(s) = \lim_{k\to\infty} \mathcal{G}_s^k\,[1](0).$$

La relation (7.9), exprimant que $\mathfrak{G}_s$ est une extension de $\mathcal{G}_s$, entraîne l'égalité $\lambda(s) = \lambda_1(s)$; Ainsi tous les opérateurs $\mathfrak{G}_s$ ont la même valeur propre dominante $\lambda(s)$.

La projection $\mathfrak{P}_s$ sur le sous-espace propre dominant peut être écrite

$$\mathfrak{P}_s[G](x_1,\dots,x_m) = E_s[G]\ \Psi_s(x_1,\dots,x_m),$$

où Ψ_s est la fonction propre dominante de $\mathfrak{G}_s$, et E_s une forme linéaire. La formule (7.9) nous permet de relier ces objets spectraux dominants des opérateurs généralisés aux objets spectraux dominants de l'opérateur de Ruelle usuel $\mathcal{G}_s$, c.-à-d., la fonction propre dominante ψ_s et le projecteur dominant e_s,

$$\Psi_s(u,u,\dots,u) = \psi_s(u), \qquad E_s[G] = e_s[g] \text{ si } g \text{ est l'application diagonale de } G.$$

Nous avons donc prouvé :

PROPOSITION 7.13. *Les opérateurs généralisés $\mathfrak{G}_s$ ont tous des propriétés spectrales dominantes. Ils partagent la même valeur propre dominante $\lambda(s)$, et leurs autres objets spectraux dominants (les fonctions propres dominantes Ψ_s et les projecteurs dominants E_s) sont des extensions des objets spectraux dominants pour l'opérateur classique $\mathcal{G}_s$ (qui n'est autre que le cas particulier de l'extension de degré $m = 1$)*

$$\Psi_s(u,\dots,u) = \psi_s(u), \qquad E_s[G] = e_s[g] \text{ où } g \text{ est l'application diagonale de } G.$$

Propriété de quasi-puissance

La série de Dirichlet $\Lambda_k^{\langle A\rangle}(F,s)$ associée au trie-tableau est un cas particulier pour lequel nous fournissons dans la suite des résultats plus précis qui seront utiles lors de l'étude de la hauteur (théorèmes 8.7 et 8.8 du chapitre 8).

PROPOSITION 7.14 (PROPRIÉTÉ DE QUASI-PUISSANCE). *Soit $\sigma > \gamma$ un réel. Pour toute distribution F associée a une densité $f \in A_\infty(\mathcal{V})$ strictement positive sur $\mathcal{V}_\mathbb{R} = \mathcal{V} \cap \mathbb{R}$, il existe une constante positive ρ telle que*

$$\rho = \lim_{k\to\infty} \frac{\Lambda_k^{\langle A\rangle}(F,\sigma)}{\lambda(\sigma)^k}. \tag{7.33}$$

Soit $\mu(\sigma)$ le module d'une valeur propre sous-dominante de $\mathfrak{G}_\sigma$, et ν une constante telle que $\nu > \mu(\sigma)$. Alors, il existe trois constantes strictement positives α, β, δ, telles que, pour tout $k \geq 1$

$$\alpha\,\lambda(\sigma)^k \leq \Lambda_k^{\langle A\rangle}(F,\sigma) \leq \beta\,\lambda(\sigma)^k, \qquad |\Lambda_k^{\langle A\rangle}(F,\sigma) - \rho\,\lambda(\sigma)^k| \leq \delta\nu^k.$$

Valeurs particulières

Pour $s = 1$, l'opérateur de Ruelle classique $\mathcal{G}_s$ est défini (puisque, par la condition (d_4) de la définition 4.1, $\gamma < 1$) et $\mathcal{G}_1$ est un transformateur de densité. Cette propriété fournit des formules explicites pour certains objets spectraux au point $s = 1$.

PROPOSITION 7.15 (VALEURS PARTICULIÈRES). *La valeur propre dominante en $s = 1$ vaut 1, le projecteur dominant e_1 de $\mathcal{G}_s$ en $s = 1$ satisfait $e_1[f] = \int_0^1 f(x)\,dx$.*

Preuve. Puisque les intervalles fondamentaux de profondeur k forment une quasi-partition de l'intervalle $\mathcal{I} = [0,1]$, on en déduit l'égalité $\Lambda_k^{\langle A\rangle}(F,1) = 1$ pour toute fonction de distribution F, et donc $\lambda(1) = 1$. De plus, l'opérateur $\mathcal{G}_1$ est un transformateur de densité : pour $f(x) > 0$ où x est réel,

$$\int_0^1 \mathcal{G}_1^k[f](t)\,dt = \int_0^1 f(t)\,dt = e_1[f] \int_0^1 \psi_1(t)\,dt + O(\rho^k),$$

permet d'obtenir $e_1[f]$, si l'on suppose que ψ_1 est définie comme une fonction de densité satisfaisant la condition de normalisation $\int_0^1 \psi_1(t)\,dt = 1$. L'expression du projecteur E_1 peut en être déduite grâce à la propriété d'extension donnée dans la proposition 7.13. Puisque l'application diagonale relative à $H_1[F]$ est exactement $F' = f$, on obtient que $E_1[H_1[F]] = e_1[f] = 1$. □

L'entropie et la probabilité de coïncidence sont définies par (4.8), (4.9) comme limites de quantités faisant intervenir des séries de Dirichlet à profondeur fixée. La propriété de quasi-puissance (7.33) fournit alors directement des expressions pour l'entropie et la probabilité à partir des objets spectraux pour $s = 1$ et $s = 2$.

PROPOSITION 7.16 (ENTROPIE ET PROBABILITÉ DE COÏNCIDENCE). *L'entropie de la source est égale à l'opposée de la dérivée de $s \mapsto \lambda(s)$ en $s = 1$, tandis que la probabilité de coïncidence est égale à $\lambda(2)$.*

7.2.4 Rayon spectral et valeur propre dominante

Du fait des propriétés spectrales dominantes, la fonction valeur propre dominante $s \mapsto \lambda(s)$ joue un rôle central dans l'analyse. Cette section établit d'importantes propriétés qui seront utilisées pour l'analyse des paramètres de trie. Tout d'abord, on s'attache à des propriétés de $\Lambda(s)$ pour s réel[3]. Ensuite, en un point s dans le demi-plan complexe $\mathrm{Re}(s) \geq \sigma$, on compare le rayon spectral $\mathfrak{R}(s)$ de $\mathfrak{G}_s$ et la valeur propre dominante $\lambda(\sigma)$ de $\mathfrak{G}_\sigma$ (qui est également la valeur propre dominante de $\mathcal{G}_\sigma$).

PROPOSITION 7.17 (PROPRIÉTÉS DE LA VALEUR PROPRE DOMINANTE). *Soit γ la constante intervenant dans la condition (d_3) de la définition 1. Les propriétés suivantes sont vraies :*
(i) *La fonction $s \mapsto \lambda(s)$ est strictement décroissante le long de l'axe réel $s > \gamma$.*
(ii) *Sur chaque droite verticale $\mathrm{Re}(s) = \sigma$, l'inégalité $\mathfrak{R}(s) \leq \lambda(\sigma)$ est vraie.*
(iii) *Dans le demi-plan $\mathrm{Re}(s) > \sigma$, les inégalités strictes $\mathfrak{R}(s) < \lambda(\sigma)$ sont vraies.*
(iv) *Si l'égalité $\mathfrak{R}(s) = \lambda(\sigma)$ est vraie pour $s = \sigma + it, t \neq 0$, alors $\mathfrak{G}_s$ a une valeur propre $\lambda = e^{ia}\lambda(\sigma)$ qui appartient au spectre de $\mathcal{G}_s$.*

Preuve. (i) La relation (7.33) permet d'exprimer la valeur propre dominante comme la limite

$$\lambda(s) = \lim_{k\to\infty} \Lambda_k^{\langle A\rangle}(1,s)^{1/k}.$$

D'après les propriétés (d_1) et (d_2) de la définition 4.1 des sources dynamiques, il existe $\delta < 1$ pour lequel $|h'(x)| \leq \delta$ pour toute branche inverse de profondeur 1, et tout réel $x \in [0,1]$. L'inégalité $|h(0) - h(1)| \leq \delta^k$ est donc vraie pour toute branche inverse de hauteur k et démontre l'inégalité

$$\lambda(s+u) \leq \delta^u\, \lambda(s) < \lambda(s).$$

La fonction $\lambda(s)$ est bien strictement décroissante sur l'axe réel.
(ii) D'après l'expression du spectre de $\mathfrak{G}_s$ donnée dans la proposition 7.7, le rayon spectral $\mathfrak{R}(s)$ de $\mathfrak{G}_s$ dépend des rayons spectraux $R(s+2\ell)$ et $\widetilde{R}(s+2\ell+1)$ des opérateurs $\mathcal{G}_s$ et $\widetilde{\mathcal{G}}_s$ pour $\ell \geq 0$. Nous commençons donc par démontrer la propriété suivante ;

LEMME 7.18. *Sur la droite $\mathrm{Re}(s) = \sigma$, le rayon spectral $\widetilde{R}(s)$ de l'opérateur $\widetilde{\mathcal{G}}_s$ et le rayon spectral $R(s)$ de l'opérateur $\mathcal{G}_s$ satisfont*

$$R(s) \leq \lambda(\sigma), \qquad \widetilde{R}(s) \leq \lambda(\sigma). \tag{7.34}$$

Preuve. Cas de base. Considérons tout d'abord le cas de $\mathcal{G}_s$. Soit s un complexe et $\sigma = \mathrm{Re}(s)$. Soit λ une valeur propre de $\mathcal{G}_s$ et $\lambda(\sigma)$ la valeur propre dominante de $\mathcal{G}_\sigma$. On considère également les vecteurs propres correspondant f et f_σ. Cette fonction f_σ est strictement positive sur le segment $\mathcal{V}_\mathbb{R}$. On normalise les fonctions f et f_σ en imposant
(i) Les deux fonctions vérifient l'inégalité

$$f_\sigma(x) \leq |f(x)| \text{ pour tout } x \in \mathcal{V}_\mathbb{R}; \tag{7.35}$$

(ii) Il existe $x_0 \in \mathcal{V}_\mathbb{R}$ tel que la borne de (i) est atteinte

$$f_\sigma(x_0) = |f(x_0)|. \tag{7.36}$$

[3] La fonction $\lambda(s)$ n'est définie que pour s réel. Grâce à la théorie de la perturbation, on peut la prolonger analytiquement dans un voisinage de l'axe réel.

On a alors

$$|\lambda\, f(x_0)| = |\mathcal{G}_s[f](x_0)| = |\sum_{i\in\mathcal{M}} \widetilde{h}_i(x_0)^s\ f\circ h_i(x_0)| \le \sum_{i\in\mathcal{M}} \widetilde{h}_i(x_0)^\sigma\ |f\circ h_i(x_0)| \tag{7.37}$$

$$\le \sum_{i\in\mathcal{M}} \widetilde{h}_i(x_0)^\sigma\ f_\sigma\circ h_i(x_0) = \lambda(\sigma) f_\sigma(x_0), \tag{7.38}$$

et la définition de x_0 prouve l'inégalité $|\lambda| \le \lambda(\sigma)$.

Cas Markovien. Pour l'opérateur $\mathcal{G}_s$ dans le cas markovien, nous considérons les mêmes objets : λ est une valeur propre de $\mathcal{G}_s$ et $f = (f_1, f_2, \dots, f_r)$ désigne un vecteur propre relatif à λ. De la même façon, le vecteur $f_\sigma = (f_{\sigma,1}, f_{\sigma,2}, \dots, f_{\sigma,r})$ désigne le vecteur propre dominant relativement à $\lambda(\sigma)$. Ce vecteur a des composantes strictement positives sur le segment $\mathcal{V}_\mathbb{R}$. De plus, on peut normaliser les vecteurs propres en imposant

(i) Pour tout $1 \le i \le r$, les composantes du vecteur propre f_i et $f_{\sigma,i}$ vérifient l'inégalité $f_{\sigma,i}(x) \le |f_i(x)|$ pour $x \in \mathcal{V}_\mathbb{R}$;

(ii) Il existe un indice ℓ et un point $x_0 \in \mathcal{V}_\mathbb{R}$ tels que $f_{\sigma,\ell}(x_0) = |f_\ell(x_0)|$ (la borne de (i) est atteinte pour une des composantes).

On a toujours

$$|\lambda\, f_\ell(x_0)| = |\sum_{j\in\mathcal{M}} \widetilde{h}_{\ell|j}(x_0)^s\ f_j\circ h_{\ell|j}(x_0)| \le \sum_{j\in\mathcal{M}} \widetilde{h}_{\ell|j}(x_0)^\sigma\ |f_j\circ h_{\ell|j}(x_0)| \tag{7.39}$$

$$\le \sum_{j\in\mathcal{M}} \widetilde{h}_{\ell|j}(x_0)^\sigma\ f_{\sigma,j}\circ h_{\ell|j}(x_0) = \lambda(\sigma) f_{\sigma,\ell}(x_0), \tag{7.40}$$

et la définition du point x_0 et de l'indice ℓ prouve l'inégalité $|\lambda| \le \lambda(\sigma)$. La propriété (7.34) est prouvée pour l'opérateur $\mathcal{G}_s$; il est clair que la preuve peut s'adapter facilement pour l'opérateur $\widetilde{\mathcal{G}}_s$. □

Revenant maintenant à l'opérateur $\mathfrak{G}_s$, la formule du spectre de la proposition 7.7 et la stricte décroissance de λ le long de l'axe réel permettent de prouver (iii) et (iv). □

7.2.5 Singularités du quasi-inverse $(I - \mathfrak{G}_s)^{-1}$ et «périodicité»

Comme il a été expliqué à la fin du chapitre 6, il est nécessaire de connaître précisément la position des pôles des différentes séries de Dirichlet $\Lambda(F, s)$. Nous rappelons que $\lambda(1) = 1$. D'après la propriété (iii) de la proposition 7.17 l'opérateur $I - \mathfrak{G}_s$ est inversible dans le demi-plan $\mathrm{Re}(s) > 1$. Ainsi, l'opérateur $(I - \mathfrak{G}_s)^{-1}$ est analytique dans ce demi-plan et admet un pôle simple en $s = 1$.

Si l'on se concentre davantage sur le comportement de cet opérateur sur la droite $\mathrm{Re}(s) = 1$, $s \ne 1$, on peut définir les *points particuliers* : ce sont les points $s = 1 + it$, avec $t \ne 0$ pour lesquelles le spectre de $\mathfrak{G}_s$ contient une valeur propre égale à 1. L'assertion (iv) de la proposition 7.17 prouve qu'en ces points particuliers, le spectre des opérateurs classiques $\mathcal{G}_s$ ou $\widetilde{\mathcal{G}}_s$ contient une valeur propre égale à 1. Les résultats suivant, qui étendent des résultats de [26], [81], [100], donnent une caractérisation des points particuliers et décrit les deux types de comportement qui peuvent être rencontrés.

PROPOSITION 7.19 (SINGULARITÉS DU QUASI-INVERSE). *L'opérateur $\mathfrak{G}_s$ a seulement deux comportements possibles sur la ligne $\mathrm{Re}(s) = 1$:*

(i) Cas apériodique. *L'opérateur $\mathfrak{G}_s$ n'a pas de points particuliers, et l'opérateur $I - \mathfrak{G}_s$ est inversible sur le demi-plan $\mathrm{Re}(s) \geq 1$ privé du point $s = 1$.*

(ii) Cas périodique. *L'opérateur $\mathfrak{G}_s$ admet des points particuliers. Ces points sont alors régulièrement espacés sur la ligne, et de la forme $1 + kit, k \in \mathbb{Z}$. L'opérateur $(I - \mathfrak{G}_s)^{-1}$ a un pôle simple en chacun de ses points particuliers, et il existe une bande à gauche de la droite $\mathrm{Re}(s) = 1$ qui ne contient pas de pôles pour cet opérateur.*

Preuve. La propriété (iv) de la proposition 7.17 montre qu'il suffit de considérer le cas de l'opérateur $\mathcal{G}_s$. L'assertion (i) est immédiate dès lors qu'il n'y a pas de points particuliers sur la droite $s = 1 + it$.

Pour (ii) supposons que $s = \sigma + it$ est un point particulier et que l'opérateur $\mathcal{G}_s$ admet $\lambda = \lambda(\sigma)$ comme valeur propre. Il y a deux cas à considérer : le cas de base et le cas markovien.

Cas de base. Nous gardons les notations utilisées dans le lemme 7.18 du paragraphe 7.2.4. La démonstration ne dépend pas de la valeur de σ mais notons que $\sigma = 1$, $\lambda = \lambda(\sigma) = 1$. Les fonctions f et f_σ sont des fonctions propres (normalisées par les conditions 7.35 et 7.36) respectivement de $\mathcal{G}_s$ et $\mathcal{G}_\sigma$ (avec $\sigma = \mathrm{Re}(s)$).

Dans un premier temps, nous montrons que f et f_σ coïncident en module sur $[0, 1]$ (i.e. $|f(x)| = f_\sigma(x)$ sur $[0, 1]$). Nous considérons le point x_0 où l'on a l'égalité

$$f_\sigma(x_0) = |f(x_0)|.$$

De cette égalité ainsi que de l'égalité $|\lambda| = \lambda(\sigma)$ nous déduisons que les inégalités (7.37), (7.38) deviennent une suite d'égalités en x_0

$$|\lambda f(x_0)| = \left| \sum_{j \in \mathcal{M}} \widetilde{h}_j(x_0)^{\sigma+it} f \circ h_j(x_0) \right| = \sum_{j \in \mathcal{M}} \widetilde{h}_j(x_0)^\sigma \, |f \circ h_j(x_0)| \tag{7.41}$$

$$= \sum_{j \in \mathcal{M}} \widetilde{h}_j(x_0)^\sigma f_\sigma \circ h_j(x_0) = \lambda(\sigma) f_\sigma(x_0). \tag{7.42}$$

Pour toute branche inverse h_j de profondeur 1, l'égalité

$$|f \circ h_j(x_0)| \; = \; f_\sigma \circ h_j(x_0)$$

est vraie, et un argument inductif prouve que la fonction que les fonctions f et f_σ sont de même module en $h_{\boldsymbol{w}}(x_0)$ pour toute branche inverse $h_{\boldsymbol{w}}$ de profondeur quelconque. Tout réel x de $[0, 1]$ étant la limite d'une suite de points $h_{\boldsymbol{w}}(x_0)$, on obtient l'égalité

$$|f(x)| = f_\sigma(x) \text{ pour tout } x \text{ dans } [0, 1].$$

Introduisons la fonction μ définie par

$$\mu(x) := \frac{f(x)}{f_\sigma(x)}, \text{ (si bien que } \mu \text{ est de module égal à 1 sur } [0, 1]). \tag{7.43}$$

Les égalités (7.41), (7.42), maintenant vraies pour tout x de $[0, 1]$, entraîne la relation

$$\sum_{j \in \mathcal{M}} \widetilde{h}_j(x)^\sigma |f \circ h_j(x)| = | \sum_{j \in \mathcal{M}} \widetilde{h}_j(x)^{\sigma+it} f \circ h_j(x)|.$$

La suite $\{a_i(x) := \widetilde{h}_i(x)^{\sigma+it} f \circ h_i(x)\}_{i\in\mathcal{M}}$ satisfait l'égalité $|\sum_{i\in\mathcal{M}} a_i(x)| = \sum_{i\in\mathcal{M}} |a_i(x)|$. Il existe donc une fonction $\theta(x)$ (de module 1) telle que

$$a_i(x) = \theta(x)|a_i(x)| \text{ pour toute branche inverse } h_i \text{ de profondeur 1.} \tag{7.44}$$

En sommant pour $j \in \mathcal{M}$, on obtient une égalité qui donne l'expression de $\theta(x)$ (en utilisant que $|f(x)| = f_\sigma(x)$ pour $x \in [0,1]$)

$$\begin{aligned}\theta(x) &= \frac{\sum_{j\in\mathcal{M}} a_i(x)}{\sum_{j\in\mathcal{M}} |a_i(x)|} \\ &= \frac{\sum_{j\in\mathcal{M}} \widetilde{h}_j(x)^{\sigma+it} f \circ h_j(x)}{\sum_{j\in\mathcal{M}} \widetilde{h}_j(x)^{\sigma} f_\sigma \circ h_j(x)} \\ &= \frac{\lambda}{\lambda(\sigma)}\mu(x).\end{aligned}$$

Puisque $\lambda = \lambda(\sigma)$, on a l'égalité $\theta(x) = \mu(x)$. De plus on calcule que pour tout $i \in \mathcal{M}$

$$\mu(x) = \frac{a_i(x)}{|a_i(x)|} = \widetilde{h}_i(x)^{it}\mu \circ h_i(x), \quad \text{pour tout } h_i \text{ de profondeur 1,} \tag{7.45}$$

et, plus généralement,

$$\widetilde{h}_{\boldsymbol{w}}(x)^{it}\mu \circ h_{\boldsymbol{w}}(x) = \mu(x), \quad \text{pour tout } h_{\boldsymbol{w}} \text{ de profondeur quelconque.} \tag{7.46}$$

Ces égalités se prolongent analytiquement sur $\mathcal{V}$ (en considérant le prolongement analytique de μ). Par conséquent, les opérateurs $\mathcal{G}_{it,h_{\boldsymbol{w}}}$ admettent tous μ comme fonction propre avec la valeur propre 1. La section 7.2.2 et l'équation (7.46) montrent que μ est une fonction propre dominante de tous les opérateurs composantes $\mathcal{G}_{it,[\boldsymbol{w}]}$ associée à la valeur propre 1.

On sait d'après la section 7.2.2 que la valeur propre dominante de $\mathcal{G}_{it,[\boldsymbol{w}]}$ est $\alpha(\boldsymbol{w})^{it}$ (avec $\alpha(\boldsymbol{w}) := \alpha(h_{\boldsymbol{w}}) = \widetilde{h}_{\boldsymbol{w}}(\underline{h_{\boldsymbol{w}}})$, $\underline{h_{\boldsymbol{w}}}$ le point fixe de $h_{\boldsymbol{w}}$ – voir la proposition 7.6). Nous en déduisons les relations

$$\alpha(\boldsymbol{w})^{it} = 1, \text{ pour tout } \boldsymbol{w} \in \mathcal{M}^*.$$

Le déterminant de Fredholm satisfait alors, d'après (7.23), la relation

$$\mathcal{F}(s+it,u) = \mathcal{F}(s,u).$$

L'opérateur admet donc des points particuliers régulièrement espacés sur la droite $s = \sigma + it$. Les propriétés du rayon spectral de $\mathcal{G}_s$ et de $\lambda(\sigma)$ assurent enfin qu'il y une bande à droite $\mathrm{Re}(s) = \sigma$ qui est libre de points particuliers[4]. *Cas markovien.* À nouveau, nous reprenons les notations du paragraphe 7.2.4. L'égalité $\lambda = \lambda(\sigma)$ transforme la suite d'inégalités (7.39), (7.40) en une suite d'égalités

$$\lambda\, f_\ell(x_0) = |\sum_j \widetilde{h}_{\ell|j}(x_0)^s\ f_j \circ h_{\ell|j}(x_0)| = \sum_j \widetilde{h}_{\ell|j}(x_0)^\sigma\ |f_j \circ h_{\ell|j}(x_0)| \tag{7.47}$$

$$= \sum_j \widetilde{h}_{\ell|j}(x_0)^\sigma\ f_{\sigma,j} \circ h_{\ell|j}(x_0) = \lambda(\sigma) f_{\sigma,\ell}(x_0). \tag{7.48}$$

[4]Voir [102] pour une caractérisation encore plus précise dans ce cas de la fonction propre des opérateurs composantes.

En particulier, pour tout symbole j, nous en déduisons l'égalité

$$|f_j \circ h_{\ell|j}(x_0)| = f_{\sigma,j} \circ h_{\ell|j}(x_0).$$

Introduisons les fonctions

$$\mu_j(x) := f_j(x)/f_{\sigma,j}(x). \tag{7.49}$$

La quantité μ_j est alors de module 1 au point $x_j := h_{\ell|j}(x_0)$. En écrivant la suite d'égalités similaire à (7.47), (7.48) mais en tenant compte de la relation $|\mu_j(x_j)| = 1$, nous obtenons

$$|f_\ell \circ h_{j|\ell}(x_j)| = f_{\sigma,\ell} \circ h_{j|\ell}(x_j),$$

et finalement, un argument inductif prouve que les quantités μ_j ($j \in \mathcal{M}$) sont de module 1 sur $[0,1]$. La suite d'égalités (7.47), (7.48)), maintenant vraies pour tout $x \in [0,1]$ et tout symbole ℓ, prouve la relation

$$\widetilde{h}_{\ell|j}(x)^{it}\ \mu_j \circ h_{\ell|j}(x) = \mu_\ell(x), \quad \text{pour tous symboles } \ell, j \text{ de } \mathcal{M},$$

Cette égalité s'étend (par prolongement analytique) à $\mathcal{V}$. En particulier

$$\widetilde{h}_{\boldsymbol{w}}(x)^{it}\ \mu_j \circ h_{\boldsymbol{w}}(x) = \mu_j(x), \quad \text{pour tout } \boldsymbol{w} \in \mathcal{C}[j]. \tag{7.50}$$

Puisque μ_j ne s'annule pas sur $[0,1]$, le paragraphe 7.2.2 et la relation (7.50) prouvent que μ_j est un vecteur propre dominant de tous les opérateurs composantes $\mathcal{G}_{it,[\boldsymbol{w}]}$, pour tout $\boldsymbol{w} \in \mathcal{C}[j]$. En particulier, μ_j est non nulle sur $\mathcal{V}$. On en déduit alors les relations

$$\alpha(\boldsymbol{w})^{it} = 1, \text{ pour } \boldsymbol{w} \in \mathcal{C},$$

où la quantité $\alpha(\boldsymbol{w})$ a été définie dans le paragraphe 7.2.2. Le déterminant de Fredholm satisfait donc, d'après la relation (7.23), l'égalité

$$\mathcal{F}(s+it, u) = \mathcal{F}(s,u).$$

□

La proposition suivante donne le développement en série de Laurent au premier ordre afin de donner une expression explicite du résidu de $(I-\mathfrak{G}_s)^{-1}$ en $s=1$ en fonction des objets spectraux dominants.

PROPOSITION 7.20 (RÉSIDU DU QUASI-INVERSE EN $s=1$). *Soit F la distribution relative à la densité f sur $[0,1]$. Alors, le résidu en $s=1$ du quasi-inverse $(I-\mathfrak{G}_s)^{-1}[H_s[F]]$ est indépendant de F et fait intervenir seulement la fonction propre dominante Ψ_s en $s=1$ sous la forme*

$$(I-\mathfrak{G}_s)^{-1}[H_s[F]] \asymp \frac{-1}{\lambda'(1)(s-1)}\Psi_1 \qquad (s=1).$$

Preuve. La théorie classique de la perturbation analytique [61] prouve l'unicité de la valeur propre dominante lorsque s est dans un voisinage suffisamment petit autour d'un point σ de l'axe des réels. Les applications $s \mapsto \lambda(s)$, $s \mapsto \Psi_s$, $s \mapsto E_s$ définissent des fonctions analytiques dans un voisinage l'axe réel. L'extension de (7.32) à un voisinage de l'axe des réels prouve que le k^{e} itéré de $\mathfrak{G}_s$ se comporte comme une puissance k^{e} de $\lambda(s)$

$$\mathfrak{G}_s^k[F](x_1,\dots,x_m) \sim \cdot\lambda(s)^k \Psi_s(F), \tag{7.51}$$

en supposant que F est positive sur $[0,1]$ et que les x_j sont tous positifs. On calcule alors facilement le résidu cherché en $s=1$. □

7.2.6 Log-concavité

L'étude asymptotique de la hauteur du trie nécessite l'utilisation d'une propriété assez fine de la valeur propre dominante de l'opérateur $\mathcal{G}_s$.

PROPOSITION 7.21 (LOG-CONCAVITÉ DE LA VALEUR PROPRE DOMINANTE). *Pour tous nombres réels positifs s, t tels que $s > t$, on a $\lambda(s)^t < \lambda(t)^s$.*

Preuve. La démonstration nécessite plusieurs étapes. Le premier lemme établit la $\log$-concavité de la fonction $\lambda(s)$.

LEMME 7.22. *Pour $1 < t < s$ La fonction $x \mapsto \lambda(x)$ est log-concave sur $[t, s]$.*

Preuve. Considérons d'abord le cas de base (c.-à-d. non markovien). Soient 2 réels s et t ($1 < t < s$) tels que la fonction $s \mapsto \lambda(s)$ est alors bien définie sur $[t, s]$. Soit également deux réels positifs α et β tels que $\alpha + \beta = 1$. On introduit la fonction

$$\Psi(x) = \frac{f_{\alpha s+\beta t}(x)}{f_s(x)^\alpha f_t(x)^\beta},$$

définie à partir de la fonction propre f_σ de $\mathcal{G}_\sigma$ pour $\sigma = \alpha s + \beta t$, $\sigma = s$, $\sigma = t$ (dont on sait que pour σ réel ce sont des fonctions strictement positives sur l'axe réel). De plus on normalise cette fonction en imposant la condition

$$\sup\{\Psi(x) \mid x \in [0, 1]\} = 1.$$

On a donc toujours pour tout $x \in [0, 1]$

$$\begin{aligned}
\lambda(\alpha s + \beta t) f_{\alpha s+\beta t}(x) &= \mathcal{G}_{\alpha s+\beta t}[f_{\alpha s+\beta t}](x) \\
&= \sum_{j\in\mathcal{M}} \widetilde{h}_j(x)^{\alpha s+\beta t} f_{\alpha s+\beta t} \circ h_j(x) \\
&\leq \sum_{j\in\mathcal{M}} \left[\widetilde{h}_j(x)^s f_s \circ h_j(x)\right]^\alpha \left[\widetilde{h}_j(x)^t f_t \circ h_j(x)\right]^\beta \\
&\leq \left(\sum_{j\in\mathcal{M}} \widetilde{h}_j(x)^s f_s \circ h_j(x)\right)^\alpha \left(\sum_{j\in\mathcal{M}} \widetilde{h}_j(x)^t f_t \circ h_j(x)\right)^\beta \\
&= \left(\mathcal{G}_s[f_s](x)\right)^\alpha \left(\mathcal{G}_s[f_t](x)\right)^\beta \\
&= \left(\lambda(s) f_s(x)\right)^\alpha \left(\lambda(t) f_s(x)\right)^\beta .
\end{aligned}$$

La première inégalité provient de la normalisation qui impose $\Psi(x) \leq 1$. La deuxième inégalité découle simplement de l'inégalité de Hölder[5]. Si $\alpha + \beta = 1$, avec $\alpha, \beta < 1$ alors

$$\sum |a_i^\alpha b_i^\alpha| \leq \left(\sum |a_i|\right)^\alpha \left(\sum |b_i|\right)^\beta .$$

Il y a égalité si $|b_i| = c\,|a_i|$. Toujours grâce à la normalisation effectuée *via* $\Psi(x)$. On a $\lambda(\alpha s + \beta t) \leq \lambda(s)^\alpha \lambda(t)^\beta$, ou encore en prenant le logarithme $\log \lambda(\alpha s + \beta t) \leq \alpha \log \lambda(s) + \beta \log \lambda(t)$. La fonction $s \mapsto \log \lambda(s)$ est bien concave sur tout intervalle réel $[t, s]$ ($1 < t < s$).

[5] En fait l'inégalité de Cauchy-Schwartz suffit ici.

Dans le cas markovien, la preuve suit les mêmes lignes, si ce n'est que l'on considère plutôt les fonctions

$$\Psi_j(x) = \frac{f_{j,\alpha s+\beta t}(x)}{f_{j,s}(x)^\alpha f_{j,t}(x)^\beta},$$

pour les j^{e} composantes des vecteurs propres f_s, f_t et f_{s+t}. Les vecteurs propres sont normalisés afin que pour tout $j \in \mathcal{M}$, $|\Psi_j(x)| \leq 1$. On a encore la chaîne d'inégalités La chaîne d'inégalités pour $i \in \mathcal{M}$

$$\begin{aligned}
\lambda(\alpha s+\beta t) f_{\alpha s+\beta t,j}(x) &= \sum_{j\in\mathcal{M}} \widetilde{h}_{j|i}(x)^{\alpha s+\beta t} f_{\alpha s+\beta t} \circ h_{i|j}(x) \\
&\leq \sum_{j\in\mathcal{M}} \left[\widetilde{h}_{j|i}(x)^s f_s \circ h_{j|i}(x)\right]^\alpha \left[\widetilde{h}(x)^t_{j|i} f_t \circ h(x)_{j|i}\right]^\beta \\
&\leq \left(\sum_{j\in\mathcal{M}} \widetilde{h}(x)^s_{j|i} f_s \circ h_{j|i}(x)\right)^\alpha \left(\sum_{j\in\mathcal{M}} \widetilde{h}(x)^t_{j|i} f_t \circ h(x)_{j|i}\right)^\beta \\
&= \left(\lambda(s) f_{s,i}(x)\right)^\alpha \left(\lambda(t) f_{s,i}(x)\right)^\beta .
\end{aligned}$$

Toujours grâce à la normalisation, cela suffit à prouver la log-concavité (large) de $\lambda(s)$ dans le cas markovien. □

Le deuxième lemme caractérise la fonction $\lambda(s)$ en cas de $\log$-concavité non stricte.

LEMME 7.23. *Si $s \mapsto \log\lambda(s)$ n'est pas strictement concave, alors l'opérateur a exactement le même spectre que celui d'un opérateur correspondant à une source sans mémoire symétrique (avec des probabilités $\frac{1}{r}$ avec $r = \text{card}\mathcal{M}$).*

Remarque. Une fonction f concave sur $[a, b]$ mais non strictement (i.e il existe $\alpha \in [0, 1]$ $f(\alpha a + (1-\alpha)b) = \alpha f(a) + (1-\alpha) f(b)$) est nécessairement affine. Donc la fonction $s \mapsto \log\lambda(s)$ est log-affine.

Preuve. On part des mêmes hypothèses que pour le lemme précédent et on définit de la même façon $\Psi(x)$. La fonction $s \mapsto \log\lambda(s)$ étant concave mais pas strictement, il existe α, β positifs tels que $\alpha + \beta = 1$ pour lesquels

$$\lambda(\alpha s+\beta t) = \lambda(s)^\alpha \lambda(t)^\beta .$$

On peut à nouveau écrire une chaîne d'inégalités de la façon suivante pour tenir compte cette fois

des branches inverses de hauteur k (et non pas de hauteur 1 comme auparavant)

$$\begin{aligned}
\lambda(\alpha s+\beta t)^k f_{\alpha s+\beta t}(x) &= \mathcal{G}_s^k[f_{\alpha s+\beta t}](x)\\
&= \sum_{\boldsymbol{w}\in\mathcal{M}^k} \widetilde{h}_{\boldsymbol{w}}(x)^{\alpha s+\beta t} f_{\alpha s+\beta t}\circ h_{\boldsymbol{w}}(x)\\
&\le \sum_{\boldsymbol{w}\in\mathcal{M}^k} \left[\widetilde{h}_{\boldsymbol{w}}(x)^s f_s\circ h_{\boldsymbol{w}}(x)\right]^\alpha \left[\widetilde{h}_{\boldsymbol{w}}(x)^t f_t\circ h_{\boldsymbol{w}}(x)\right]^\beta\\
&\le \left(\sum_{\boldsymbol{w}\in\mathcal{M}^k} \widetilde{h}_{\boldsymbol{w}}(x)^s f_s\circ h_{\boldsymbol{w}}(x)\right)^\alpha \left(\sum_{\boldsymbol{w}\in\mathcal{M}^k} \widetilde{h}_{\boldsymbol{w}}(x)^t f_t\circ h_{\boldsymbol{w}}(x)\right)^\beta\\
&= \left(\mathcal{G}_s^k[f_s](x)\right)^\alpha \left(\mathcal{G}_s^k[f_t](x)\right)^\beta\\
&= \left(\lambda(s)^k f_s(x)\right)^\alpha \left(\lambda(t)^k f_s(x)\right)^\beta.
\end{aligned}$$

L'inégalité valable pour tout $x\in[0,1]$

$$\lambda(\alpha s+\beta t)^k f_{\alpha s+\beta t}(x) \le \left(\lambda(s)^k f_s(x)\right)^\alpha \left(\lambda(t)^k f_s(x)\right)^\beta$$

devient une égalité si on choisit x_0 tel que $\Psi(x_0)=1$. Il s'ensuit que toute la chaîne d'inégalités devient une chaîne d'égalités pour ce choix ($x=x_0$). Ainsi, de même qu'auparavant on a l'égalité

$$\sum_{\boldsymbol{w}\in\mathcal{M}^k} \widetilde{h}_{\boldsymbol{w}}(x)^{\alpha s+\beta t} f_{\alpha s+\beta t}\circ h_{\boldsymbol{w}}(x_0) = \sum_{\boldsymbol{w}\in\mathcal{M}^k} \left[\widetilde{h}_{\boldsymbol{w}}(x)^s f_s\circ h_{\boldsymbol{w}}(x_0)\right]^\alpha \left[\widetilde{h}_{\boldsymbol{w}}(x)^t f_t\circ h_{\boldsymbol{w}}(x_0)\right]^\beta,$$

mais cette fois on considère toutes les branches inverses de de profondeur k fixée. Pour tout $h_{\boldsymbol{w}}$ de hauteur quelconque $|\boldsymbol{w}|$, on a donc

$$\Psi(h_{\boldsymbol{w}}(x_0)) = 1,$$

ce qui prouve

$$\Psi(x) = 1 \quad \text{pour tout } x\in[0,1].$$

L'inégalité de Hölder devient également une égalité donc la quantité

$$c(|\boldsymbol{w}|,x) = \frac{\widetilde{h}_{\boldsymbol{w}}(x)^s f_s\circ h_{\boldsymbol{w}}(x)}{\widetilde{h}_{\boldsymbol{w}}(x)^t f_t\circ h_{\boldsymbol{w}}(x)} = \widetilde{h}_{\boldsymbol{w}}(x)^{s-t}\frac{f_s\circ h_{\boldsymbol{w}}(x)}{f_t\circ h_{\boldsymbol{w}}(x)}$$

ne dépend que de x et de $|\boldsymbol{w}$, la hauteur de la branche inverse considérée (ou encore la longueur du préfixe $\boldsymbol{w}$), et non pas de la branche elle-même. En posant $\Delta_{s,t}(x)=\frac{f_s(x)}{f_t(x)}$, on a l'expression équivalente

$$c(|\boldsymbol{w}|,x) = \widetilde{h}_{\boldsymbol{w}}(x)^{s-t}\Delta_{s,t}\circ h_{\boldsymbol{w}}(x) = \mathcal{G}_{s-t}, h_{\boldsymbol{w}}[\Delta_{s,t}](x).$$

Soit $\boldsymbol{v}=\boldsymbol{w}^k$. Avec $h_{\boldsymbol{w}}$ et $h_{\boldsymbol{v}}$ les branches inverses associées, on écrit

$$c(|\boldsymbol{v}|,x) = \mathcal{G}^k_{s-t,h_{\boldsymbol{w}}}[\Delta_{s,t}](x) = \widetilde{h}_{\boldsymbol{v}}(x)^{s-t}\Delta_{s,t}\circ h_{\boldsymbol{v}}(x) = c(k|\boldsymbol{w}|,x).$$

Or comme $\Delta_{s,t}$ est strictement positive sur $[0,1]$, on a aussi la relation suivante concernant le spectre de $\mathcal{G}_{s,h}$ (en prenant compte de la valeur propre dominante)

$$(\forall x\in[0,1])\quad \alpha(h_{\boldsymbol{w}})^{s-t} = \lim_{k\to\infty}\left(\mathcal{G}^k_{s-t,h_{\boldsymbol{w}}}[\Delta_{s,t}](x)\right)^{1/k} = \lim_{k\to\infty} c(k|\boldsymbol{w}|,x)^{1/k}.$$

On en déduit que pour tous les préfixes $\boldsymbol{w}$ de même longueur ρ la quantité $\alpha(\boldsymbol{w})$ dépend uniquement de ρ. De plus, on peut calculer explicitement $\alpha(\boldsymbol{v})$ pour $\boldsymbol{v} = i^k$ (avec i une lettre de l'alphabet $\mathcal{M}$). En effet

$$\alpha(\boldsymbol{v}) = \widetilde{h}_{\boldsymbol{v}}(z^\star),$$

où $\widetilde{h}_{\boldsymbol{v}}$ est le prolongement analytique de $|h_s e q v'|$ et $z^\star$ est l'unique point fixe de $h_{\boldsymbol{v}}$ sur $[0,1]$. On remarque immédiatement que $z^\star$ est également point fixe de h_i (et par suite de tous ses itérés). En effet, si on considère z_0 le point fixe unique de h_i, $h_i(z_0) = z_0$ entraîne $h_{\boldsymbol{v}}(z_0) = h_i^k(z_0) = z_0$. Donc comme $z^\star$ est unique, on a $z^\star = z_0$. On peut alors écrire

$$(h_i^k(z^\star))' = \prod_{m=0}^{k-1} h_i' \circ h_i^m(z^\star) = h_i'(z^\star)^k.$$

On en déduit donc que $\alpha(\boldsymbol{v}) = \alpha^k$, où α est la valeur de $\alpha(h_i)$ pour toute branche inverse h_i de hauteur 1.

Dans le cas de l'alphabet fini de cardinal r, on peut encore préciser cette constante α. En effet, on a pour tout x,r étant également le nombre de branches inverses de hauteur 1,

$$\begin{aligned}\mathcal{G}_{s-t}^k[\Delta_{s,t}](x) &= \sum_{\boldsymbol{w}\in\mathcal{M}^k} \widetilde{h}_{\boldsymbol{w}}(x)^{s-t} \Delta_{s,t} \circ h_{\boldsymbol{w}}(x) \\ &= \sum_{\boldsymbol{w}\in\mathcal{M}^k} c(k,x) \\ &= r^k c(k,x).\end{aligned}$$

En tenant compte du comportement dominant de $\mathcal{G}_s$, on obtient pour tout $x \in [0,1]$

$$\lambda(s-t) = \lim_{k\to\infty} \left(\mathcal{G}_{s-t}^k[\Delta_{s,t}](x)\right)^{1/k} = \lim_{k\to\infty} \left(r^k c(k,x)\right)^{1/k} = r\alpha^{s-t}.$$

Donc on a pour s réel, l'égalité

$$\lambda(s) = r\alpha^s.$$

De plus, l'égalité $\lambda(1) = 1$ (valable pour des raisons générales ; cela correspond au cas du transformateur de densité) permet de déterminer

$$\alpha = \frac{1}{r}.$$

On en déduit l'égalité des spectres avec un opérateur correspondant à une source sans mémoire symétrique [6] B (associé aux probabilités uniformes $\frac{1}{r}$) grâce à la formule de la trace. Enfin pour s réel, on obtient

$$\lambda(s) = r^{1-s}.$$

En particulier, $\log\lambda(s) = (1-s)\log r$ est affine.

La preuve s'adapte au cas markovien. Si la valeur $\lambda(s)$ propre est log-affine, alors la source est constituée de r sources sans mémoire symétriques. □

[6] J'écris ici «un» opérateur, car il y en a 2^r (correspondant pour chaque branche affine à une pente positive ou négative).

On peut maintenant démontrer la propriété de log-concavité. Soit $1 < t < s$, montrons qu'on a toujours l'inégalité stricte

$$\lambda(s)^t < \lambda(t)^s.$$

D'après le lemme 7.22, la fonction $s \mapsto \log(\lambda(s))$ est concave. La preuve distingue donc 2 cas, selon que la concavité est stricte ou large.

(i) *la fonction $s \mapsto \log(\lambda(s))$ est strictement concave.* Il existe $k \in \mathbb{N}$ tel que $s \in [t, kt]$. Soit α et β tels que $s = \alpha t + \beta kt$ et $\alpha + \beta = 1$. La stricte concavité de f entraîne

$$f(s) < \alpha f(t) + \beta f(kt), \text{ ou encore } \lambda(s) < \lambda(t)^\alpha {\lambda_{kt}}^\beta.$$

On sait que pour t réel et k entier on a $\lambda_{kt} \leq \lambda(t)^k$. Élevant à la puissance t, on obtient

$$\lambda(s)^t < \left(\lambda(t)^\alpha {\lambda_t}^{k\beta}\right)^t = \lambda(t)^{t(\alpha+\beta k)} = \lambda(t)^s,$$

et finalement

$$\lambda(s)^t < \lambda(t)^s.$$

(ii) *la fonction $s \mapsto \log \lambda(s)$ n'est pas strictement concave.* Par conséquent, cette fonction est affine (voir le remarque après le lemme 7.23)

$$\log \lambda(s) = (1 - s) \log r,$$

où r est le nombre de branches de hauteur 1. Pour $1 < t < s$, on a facilement

$$t \log \lambda(s) = t(1 - s) \log r < s(1 - t) \log r = s \log \lambda(t).$$

On conclut que l'inégalité

$$\lambda(s)^t < \lambda(t)^s$$

est toujours vraie pour $1 < t < s$. □

Ceci clôt l'étude des propriétés des opérateurs de Ruelle généralisés. Ces propriétés interviennent dans la détermination du comportement asymptotique des paramètres de tries généraux construits sur des mots émis par une source dynamique probabilisée.

Chapitre 8

Analyse des tries généraux

Sommaire

Nous pouvons maintenant retourner à l'analyse des paramètres de trie en commençant avec les paramètres additifs de taille et de longueur de cheminement (avec les variantes liées à l'hybridation). Nous introduisons d'abord les principaux outils, la transformée de Mellin, qui relie les pôles de la série de Dirichlet et le comportement asymptotique des sommes harmoniques obtenues pour les paramètres additifs (taille et longueur de cheminement). Nous obtenons alors les résultats principaux dans le modèle de Poisson. L'étape suivante consiste en une étape de «dépoissonisation de Dirichlet» qui donne les comportements asymptotiques correspondant dans le modèle de Bernoulli.

Dans le modèle de Poisson, N est fortement concentré autour de sa moyenne z avec une grande probabilité, si bien que le paramètre z joue un rôle tout à fait similaire à celui du paramètre n dans le modèle de Bernoulli. Il est alors raisonnable d'espérer que les valeurs moyennes de paramètres dans les deux modèles sont asymptotiquement équivalents. La dépoissonisation asymptotique est un moyen d'établir de telles équivalences, en supposant des hypothèses supplémentaires.

1. *Dépoissonisation de Dirichlet.* Elle repose sur l'existence de transformées de Mellin et d'un principe d'approximation entre les séries de Dirichlet dans les modèles de Poisson et de Bernoulli. Cette technique est utilisée dans cette thèse pour l'analyse en moyenne des paramètres additifs de taille et de longueur de cheminement dans le modèle de Bernoulli : le théorème 8.5 dérive du théorème 8.2, établi dans le modèle de Poisson, de cette façon.
2. *Dépoissonisation par méthode de col.* Cette technique repose sur des estimations de valeurs moyennes dans le modèle de Poisson (ou, de façon équivalente, de séries génératrices) pour des valeurs du paramètre z prise dans le plan complexe, en conjonction avec une analyse du point de col. Cette technique est utilisée pour l'étude de la distribution du paramètre multiplicatif de

la hauteur dans le modèle de Bernoulli : Le théorème 8.8 dérive ainsi de la version de Poisson (théorème 8.7).

Le résultat principal ici est que le comportement asymptotique dominant dans le modèle de Poisson reste valide dans le modèle de Bernoulli.

8.1 Transformée de Mellin

La transformée de Mellin est la méthode de choix pour obtenir le développement asymptotique de sommes harmoniques. Elle est constamment utilisée pour étudier le comportement asymptotique des tries. Nous rappelons les propriétés déjà évoquées dans le cadre classique au chapitre 2 (voir la monographie [37] pour un traitement extensif). La transformée de Mellin l'application définie par

$$g^*(s) := \int_0^\infty g(x)\, x^{s-1}\, dx.$$

Elle est définie dans la bande $\alpha < \mathrm{Re}(s) < \beta$ appelée bande fondamentale qui est notée $\langle \alpha, \beta \rangle$. Par exemple, la transformée de Mellin de e^{-x} est la fonction gamma d'Euler $\Gamma(s)$ et la bande fondamentale associée est $\langle 0, +\infty \rangle$.

Une somme harmonique est une somme de la forme

$$G(x) = \sum_{k \in K} \lambda_k g(\mu_k x), \tag{8.1}$$

où les coefficients $\{\lambda_k\}$ et $\{\mu_k\}$ sont appelés amplitudes et fréquences, et la fonction g est la fonction de base. L'analyse des tries se réduit essentiellement à l'estimation asymptotique de sommes harmoniques particulières.

Deux principes de l'analyse de Mellin sont utilisées ici (voir [37] pour un traitement détaillé).

(M_1) *Propriété de factorisation des sommes harmoniques.* La transformée d'une somme harmonique définie par (8.1) satisfait

$$G^*(s) = \Xi(s) \cdot g^*(s), \quad \text{avec } \Xi(s) = \sum_{k \in K} \lambda_k \mu_k^{-s},$$

dans l'intersection des bandes de convergence absolue de $\Xi(s)$ et de $g^*(s)$.

(M_2) *Propriété de correspondance.* La transformée de Mellin fait correspondre les termes du développement asymptotique de la fonction $G(x)$ (ici, typiquement une somme harmonique issue de (M_1) avec les singularités de la transformée de Mellin $G^*(s)$. La correspondance est vraie dans les deux sens et se résume en la règle suivante : Supposons que $G^*(s)$ définie sue une bande $\langle \alpha, \beta \rangle$ admette un prolongement méromorphe sur la bande étendue $\langle \alpha, \gamma \rangle$ avec $\gamma \geq \beta$, soit analytique sur la droite $\mathrm{Re}(s) = \gamma$, et satisfasse $G^*(s) = O(|s|^{-r})$ pour $r > 1$ lorsque $\mathrm{Im}(s) \to \pm\infty$ dans $\alpha \leq \mathrm{Re}(s) \leq \gamma$. Alors chaque terme singulier dans l'expansion locale de $G^*(s)$ en un pôle de la bande étendue donne un terme correspondant du développement asymptotique de $G(x)$ en $+\infty$, selon la traduction :

$$\left(\frac{d}{(s-\xi)^k} \right) \Longrightarrow \left(\frac{(-1)^k d}{(k-1)!} x^{-\xi} (\log x)^{k-1} \right), \tag{8.2}$$

avec un terme d'erreur en

$$O(x^{-\gamma}). \tag{8.3}$$

L'application jointe des deux règles (M_1) and (M_2) donne le principe de base de l'analyse asymptotique de fonctions harmoniques grâce à la transformée de Mellin :

> *La transformée de Mellin factorise une somme harmonique $G(x)$, de telle façon que le développement asymptotique de la somme se ramène à l'analyse séparée de deux ensembles de singularités : celles de la transformée de Mellin de la fonction de base $g^*(s)$ et celles de la série de Dirichlet associée $\Xi(s)$.*

L'analyse de Mellin de sommes harmoniques est souvent conduite dans des situations où : (i) la transformée g^* de la fonction de base est de décroissance exponentielle (ce qui est systématiquement le cas dès lors que la fonction exponentielle entre en jeu) ; (ii) la série de Dirichlet $\Xi(s)$ est au plus de croissance polynomiale lorsque $|\mathrm{Im}(s)| \to \infty$. Dans ce cas, la transformée $G^*(s)$ est de décroissance rapide en $\pm i\infty$ et l'application de (8.2) est légitime.

L'examen de (8.2) révèle que des pôles complexes entraîne des fluctuations périodiques. Le cadre d'application s'étend (voir [37]) à toute fonction qui a une infinité de pôles dans une bande finie, à condition que la fonction reste de croissance polynomiale $O(|s|^{-r})$ avec $r > 1$ dans la bande sur un ensemble de parallèles à l'axe des réels mais s'en éloignant vers $\pm i\infty$. On parle alors de croissance polynomiale dans le sens «faible» lorsque la fonction ne vérifie la condition de croissance polynomiale que sur une telle «échelle» verticale. Dans ce cas, il y a superposition de fluctuations périodiques qui, collectivement, peuvent ou non être périodiques selon que les pôles disposés sur une droite verticale sont espacés régulièrement.

Lorsque les pôles s'accumulent à droite de la frontière $\mathrm{Re}(s) = \gamma$ de la bande $\langle \alpha, \gamma \rangle$, le paradigme général de (8.2) et (8.3) nécessite une «retouche». En effet, il n'est pas immédiatement clair que les fluctuations induites contribuent globalement en un terme qui reste $o(x^{-\gamma})$. Or cette situation apparaît naturellement dans le contexte des sources dynamiques, lorsqu'il y a accumulation de pôles à droite de la frontière $\mathrm{Re}(s) = \gamma$.

PROPOSITION 8.1 (ASYMPTOTIQUE «AMÉLIORÉE» DE MELLIN). *Supposons que la transformée de Mellin $G^*(s)$ de $G(x)$ définie dans $\langle \alpha, \beta \rangle$ admette un prolongement méromorphe dans la bande $\langle \alpha, \gamma \rangle$ avec $\gamma \geq \beta$, satisfasse $O(|s|^{-r})$ avec $r > 1$ dans le sens «faible» dans $\alpha \leq \mathrm{Re}(s) \leq \gamma$, et soit méromorphe sur la droite $\mathrm{Re}(s) = \gamma$ avec seulement un nombre fini de pôles sur cette droite. Alors on a*

$$G^*(s) \asymp \sum_{(\xi,k)} \frac{d_{\xi,k}}{(s-\xi)^k} \quad \Longrightarrow \quad G(x) = \sum_{(\xi,k)} \frac{(-1)^k d_{\xi,k}}{(k-1)!} x^{-\xi} (\log x)^{k-1} + o(x^{-\gamma}),$$

lorsque $x \to +\infty$, où ξ parcourt l'ensemble des pôles tels que $\alpha \leq \mathrm{Re}(\xi) \leq \gamma$.

L'expression singulière d'une fonction $\Omega(s)$ dans un domaine est dénotée par '$\asymp$'. C'est la somme formelle des expressions singulières locales de $\Omega(s)$ en chaque singularité du domaine [37].

Preuve. En soustrayant des fonctions élémentaires de $G(x)$, nous pouvons supposer sans perte de généralité que $G(x)$ est analytique sur $\mathrm{Re}(s) = \gamma$. (des combinaisons bien choisies d'exponentielles et de monômes en x par exemple, puisque leurs transformées sont de décroissance rapide en $\pm i\infty$.) La preuve classique de (8.2) dans (M_2) se fait en intégrant sur un large rectangle avec des cotés verticaux $\mathrm{Re}(s) = \alpha$ et $\mathrm{Re}(s) = \gamma$; voir [37]. Il faut donc prouver que sur la frontière droite,

$$\int_{\gamma - i\infty}^{\gamma + i\infty} G^*(s) x^{-s}\, ds = o(x^{-\gamma}).$$

Posons $s = \gamma + iu$ et $x = e^y$. Alors, l'équivalent

$$x^{-\gamma} \int_{-\infty}^{+\infty} G^*(\gamma + iu) e^{-iuy}\, du = o(x^{-\gamma})$$

est vérifié par application du lemme de Riemann-Lebesgue : pour toute fonction $h(u)$ appartenant à $\mathcal{L}^1(-\infty, +\infty)$, on a

$$\int_{-\infty}^{+\infty} h(u) e^{iuy}\, du = o(1),$$

lorsque $y \to \pm\infty$. □

8.2 Taille et de la longueur de cheminement (Poisson)

L'analyse de la taille et de la longueur de cheminement est différente selon que l'on considère un alphabet fini ou infini.

8.2.1 Taille et longueur de cheminement (alphabet fini)

D'après la proposition 7.19 de la section 7.2.5, l'opérateur $I - \mathfrak{G}_s^{\langle A\rangle}$ est inversible dans le demi-plan $\mathrm{Re}(s) > 1$. La série $\Lambda^{\langle A\rangle}(F, s)$ y est donc analytique et admet un pôle simple en $s = 1$. Selon la périodicité de la source, deux types de résultats doivent être distingués :

(a) Dans le cas périodique, $\Lambda(F, s)$ admet des pôles sur la droite $\mathrm{Re}(s) = 1$ qui sont espacés régulièrement sur cette droite et contribuent par des termes périodiques au développement asymptotique. De plus, il existe une bande verticale $\langle 1 - \delta, 1\rangle$ (avec $0 < \delta < 1$) dénuée de pôles. Cela fournit de bonnes bornes sur les termes d'erreurs.

(b) Dans le cas apériodique, il n'y a pas d'autres pôles sur la droite $\mathrm{Re}(s) = 1$. Cependant, il peut y avoir une accumulation de pôles à gauche de la droite $\mathrm{Re}(s) = 1$. La Proposition 8.1 permet d'évaluer alors directement la contribution de tels pôles.

THÉORÈME 8.2. (PARAMÈTRES ADDITIFS DANS LE MODÈLE DE POISSON : ALPHABET FINI). *Soit $(\mathcal{P}_z, \mathsf{S}, F)$ le modèle de Poisson de paramètre z relatif à une source S avec un alphabet fini et une distribution initiale F.*

(i) Dans le cas où la source est apériodique, les valeurs moyennes de la taille et de la longueur de cheminement d'un trie sont

$$\widehat{S}(z) = \frac{1}{h(\mathsf{S})} z + o(z), \qquad \widehat{P}_A(z) = \frac{1}{h(\mathsf{S})} z \log z + z(C_A(F, \mathsf{S}\frac{\gamma}{h(\mathsf{S})}) + o(z).$$

Les espérances de la longueur de cheminement des tries-liste et tries-abr sont de la forme

$$\widehat{P}_N(z) = \frac{K_N(\mathsf{S})}{h(\mathsf{S})} z \log z + (C_N(F, \mathsf{S}) + \frac{\gamma}{h(\mathsf{S})}) z + o(z),$$

où N peut être B ou L pour les versions en arbres binaires de recherche (abr) et en listes.

(ii) Dans le cas où la source est périodique, les espérances de la taille et de la longueur de cheminement d'un trie sont

$$\widehat{S}(z) = \frac{1}{h(\mathsf{S})} z\,[1 + Q_A(\log z)] + o(z^{1-\delta})$$

$$\widehat{P}_A(z) = \frac{1}{h(\mathsf{S})} z \log z + z\,[C_A(F, \mathsf{S}) + \frac{\gamma}{h(\mathsf{S})} + Q_A(\log z)] + o(z^{1-\delta}).$$

Les espérances de la longueur de cheminement pour les tries-liste et tries-abr sont de la forme (où N peut être B ou L)

$$\widehat{P}_N(z) = \frac{K_N(\mathsf{S})}{h(\mathsf{S})} z\log z + z\,[C_N(F,\mathsf{S}) + \frac{\gamma}{h(\mathsf{S})} + Q_N(\log z)] + o(z^{1-\delta}).$$

La fonction $Q_N(u)$ dépend de la source S et est de très faible amplitude ; δ est une constante positive, satisfaisant $0 < \delta < 1$, qui est déterminée par la région sans pôles de $\Lambda(F,s)$ à gauche de la droite $\mathrm{Re}(s) = 1$. Les constantes $K_N(\mathsf{S})$ ne dépendent pas de la distribution initiale contrairement aux constantes $C_N(F,\mathsf{S})$.

Preuve. Les expressions données dans le théorème 6.14 sont des sommes harmoniques où la fonction exponentielle est présente. La transformée de Mellin de e^{-x} est la fonction gamma $\Gamma(s)$ qui possède des singularités en chaque entier négatif. Les séries de Dirichlet de ces sommes harmoniques sont de la forme $\Xi(s) = \Lambda^{\langle\cdot\rangle}(F,-s)$, où $\Lambda^{\langle\cdot\rangle}(F,s)$ est la série de Dirichlet des mesures fondamentales associée.

De façon plus précise, les expressions du théorème 6.14 admettent les transformées de Mellin suivantes (les bandes fondamentales sont également précisées dans chaque cas) :
– Taille.

$$\mathsf{mellin}(\widehat{S}(z),s) = -\Lambda^{\langle A\rangle}(F,-s)(s+1)\Gamma(s), \qquad s \in \langle -2,-1\rangle\,.$$

– Longueur de cheminement.

$$\mathsf{mellin}(\widehat{P}_A(z),s) = -\Lambda^{\langle A\rangle}(F,-s)\Gamma(s+1), \qquad s \in \langle -2,-1\rangle\,.$$

– Longueur de cheminement :

$$\mathsf{mellin}(\widehat{P}_L(z),s) = -\Lambda^{\langle L\rangle}(F,-s)\Gamma(s+1), \qquad s \in \langle -2,-1\rangle \qquad \text{pour les tries-listes}$$
$$\mathsf{mellin}(\widehat{P}_B(z),s) = -\Lambda^{\langle B\rangle}(F,-s)\Gamma(s), \qquad s \in \langle -2,-1\rangle \qquad \text{pour les tries-abr.}$$

Toutes ces transformées se prolongent en des fonctions méromorphes dans une bande plus large à droite de la droite $\mathrm{Re}(s) = -1$ de façon convenable pour l'analyse de Mellin.

Concernant la condition de croissance, la fonction $\Gamma(s)$ est de décroissance exponentielle

$$|\Gamma(\sigma+it)| \sim \sqrt{2\pi}\,|t|^{\sigma-1/2}\,e^{-\pi|t|/2} \qquad (t \to +\infty), \tag{8.4}$$

tandis que les différentes séries de Dirichlet $\Lambda^{\langle\cdot\rangle}(F,-s)$ sont de croissance modérée sur toute bande verticale $\langle -2,-a\rangle$ avec $a > 1$. Par exemple, la série $\Lambda^{\langle A\rangle}(F,s)$ admet la formule intégrale suivante, vraie pour $\mathrm{Re}(s) > 1$:

$$\Lambda^{\langle A\rangle}(F,s) = s\int_0^\infty W(x)e^{-sx}dx, \quad \text{avec } W(x) = \sum_{u_{\boldsymbol{w}} \geq e^{-x}} 1.$$

Cela entraîne, pour $\mathrm{Re}(s) = \sigma < -1$,

$$|\Lambda^{\langle A\rangle}(F,-s)| \leq |s| \int_0^{+\infty} W(x)e^{\sigma x} = |s|\,\frac{\Lambda^{\langle A\rangle}(F,-\sigma)}{\sigma},$$

qui est de croissance modérée $O(|s|)$ sur la bande finie $\langle -2,-a\rangle$ ($a > 1$).

À partir des propriétés de $\mathfrak{G}_s$, l'expression singulière de $\Lambda^{\langle A\rangle}(F,s)$ en $s=1$ s'écrit

$$\Lambda^{\langle A\rangle}(F,s) \asymp \frac{-1}{\lambda'(1)}\,\frac{1}{s-1} + C_A(F,\mathsf{S}) \qquad (s=1).$$

Dans le cas des tries hybrides (en liste ou en abr), les expressions (7.18), (7.17) dans le cas de base, (7.21), (7.22) pour le cas Markovien, en prenant compte des valeurs particulières données dans la proposition 7.15, impliquent, pour l'expression singulière de $\Lambda^{\langle N\rangle}(F,s)$ en $s=1$

$$\Lambda^{\langle N\rangle}(F,s) \asymp \frac{-1}{\lambda'(1)}\,\frac{K_N(\mathsf{S})}{s-1} + C_N(F,\mathsf{S}) \qquad (s=1). \tag{8.5}$$

Les constantes peuvent être explicitées en fonctions des objets spectraux dominants de $\mathfrak{G}_s$ en $s=1$. Si l'on reprend la notation utilisée en (7.32)

$$\mathfrak{G}_s^k = \lambda(s)^k\,\mathfrak{P}_s + \mathfrak{N}_s^k$$

où $\mathfrak{P}_s$ est la projection sur le sous-espace propre dominant, et $\mathfrak{N}_s$ est un opérateur relatif au reste du spectre (l'opérateur $\mathfrak{N}_s$ a donc un rayon spectral strictement inférieur à la valeur propre dominante $\lambda(s)$). La projection $\mathfrak{P}_s$ sur le sous-espace dominant s'écrit encore

$$\mathfrak{P}_s[G](x_1,\dots,x_m) = E_s[G]\;\Psi_s(x_1,\dots,x_m),$$

où Ψ_s est la fonction propre dominante de $\mathfrak{G}_s$, et E_s une forme linéaire. Les opérateurs interviennent avec la valeur $s=1$ et on sait d'après la proposition 7.15 que $E_1[H_1^{\langle N\rangle}[F]=1$.
L'équation (8.5) peut être davantage précisée. Les deux premiers termes de l'expression singulière pour $(I-\mathfrak{G}_s)^{-1}$ en $s=1$ sont

$$(I-\mathfrak{G}_s)^{-1}[G] \asymp \frac{\Psi_1}{1-\lambda(s)} + (I-\mathfrak{N}_1)^{-1}[G], \qquad (s=1).$$

Les constantes $K_N(\mathsf{S})$ dépendent uniquement des «mesures fondamentales uniformes» de profondeur 1, notées $u_i^\star$ et de la valeur de la fonction propre dominante Ψ_1 aux extrémités des intervalles fondamentaux de profondeur 1.

Note. Pour le cas le plus simple (trie ordinaire) on a

$$\mathfrak{G}_1^{\langle A\rangle\,k} = \Psi^{\langle A\rangle}(0,1) + \mathfrak{N}_1^k(0,1) = \sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}} = 1,$$

ce dont on déduit $\Psi^{\langle A\rangle}(0,1)=1$ et $\mathfrak{N}_1^{\langle A\rangle}=0$.
Par exemple, dans le cas d'une source dynamique de base avec un codage monotone (croissant ou décroissant), on a

$$K_A(\mathsf{S}) = 1,$$

$$K_L(\mathsf{S}) = \sum_i U^\star_{[>i]}\;\Psi_1^{\langle L\rangle}(h_i^-,h_i^+,h_r^+), \qquad K_B(\mathsf{S}) = 2\sum_{i<j}\frac{u_i^\star\,u_j^\star}{U^\star_{[i,j]}}\;\Psi_1^{\langle B\rangle}(h_i^-,h_i^+,h_j^-,h_j^+).$$

Les fonctions $L_1^{\langle L\rangle}$ et $L_1^{\langle B\rangle}$ sont définies à l'aide de la fonction de densité F

$$L_1^{\langle L\rangle}(x_1,x_2,x_3) = \left|\frac{F(x_2)-F(x_3)}{x_2-x_3}\right|$$

$$L_1^{\langle B\rangle}(x_1,x_2,x_3,x_4) = \left|\frac{F(x_1)-F(x_2)}{x_1-x_2}\right|\cdot\left|\frac{F(x_3)-F(x_4)}{x_3-x_4}\right|\cdot\left|\frac{x_1-x_4}{F(x_1)-F(x_4)}\right|.$$

Les réels h_i^+ et h_i^- pour $i \in \mathcal{M}$ dénotent les extrémités de l'intervalle fondamental de niveau 1 correspondant à la branche inverse h_i (et sont définis à l'équation (7.16) pour un codage croissant ou décroissant).

Les constantes $C_N(\mathsf{S})$ font de plus rentrer en compte les objets spectraux sous-dominants pour $s = 1$.

$$C_A(\mathsf{S}, F) = \frac{\lambda''(1)}{(\lambda'(1))^2} + 1$$

$$C_L(\mathsf{S}, F) = \frac{\lambda''(1)}{(\lambda'(1))^2} \sum_i U^\star_{[>i]} \Psi_1^{\langle L\rangle}(h_i^-, h_i^+, h_r^+) + (I - \mathfrak{N}_1^{\langle L\rangle})^{-1}\, [L_1^{\langle L\rangle}](h_i^-, h_i^+, h_r^+)$$

$$C_B(\mathsf{S}, F) = 2\frac{\lambda''(1)}{(\lambda'(1))^2} \sum_{i<j} \frac{u_i^\star\, u_j^\star}{U^\star_{[i,j]}} \Psi_1^{\langle B\rangle}(h_i^-, h_i^+, h_j^-, h_j^+)$$
$$+ 2\sum_{i<j} \frac{u_i^\star\, u_j^\star}{U^\star_{[i,j]}}\, (I - \mathfrak{N}_1^{\langle B\rangle})^{-1} [L_1^{\langle B\rangle}](h_i^-, h_i^+, h_j^-, h_j^+).$$

Dans le cas Markovien, les formules sont similaires mais doivent prendre en compte les composantes du vecteur propre Ψ_1. (voir les équations (7.21) et (7.22)). Par exemple pour le trie ABR, on a

$$K_B(\mathsf{S}) = 2\sum_{\substack{\ell,i,j\\ i<j}} \frac{u^\star_{i|\ell} u^\star_{j|\ell}}{U^\star_{[i,j]|\ell}} {}^t\mathbf{e}_\ell \Psi_1^{\langle B\rangle}(h^-_{i|\ell}, h^+_{i|\ell}, h^-_{j|\ell}, h^+_{j|\ell}),$$

$$C_B(\mathsf{S}, F) = 2\frac{\lambda''(1)}{(\lambda'(1))^2} \sum_{\substack{\ell,i,j\\ i<j}} \frac{u^\star_{i|\ell} u^\star_{j|\ell}}{U^\star_{[i,j]|\ell}} \frac{\lambda''(1)}{(\lambda'(1))^2} {}^t\mathbf{e}_\ell \Psi_1^{\langle B\rangle}(h_i^-, h_i^+, h_j^-, h_j^+)$$
$$+ 2\sum_{\substack{\ell,i,j\\ i<j}} \frac{u^\star_{i|\ell} u^\star_{j|\ell}}{U^\star_{[i,j]|\ell}}\, (I - {}^t\mathbf{e}_\ell \mathfrak{N}_1^{\langle B\rangle})^{-1} [L_1^{\langle B\rangle}](h_i^-, h_i^+, h_j^-, h_j^+).$$

Par ailleurs la fonction $\Gamma(s)$ admet les expressions singulières

$$\Gamma(s) \asymp \frac{-1}{s+1} + \gamma - 1 \quad (s = -1), \qquad \Gamma(s+1) \asymp \frac{1}{s+1} - \gamma \quad (s = -1).$$

La preuve peut être complétée grâce aux équations (7.18) et (7.17) combinées avec un calcul de résidu. □

Dans le cas d'une source périodique, les fluctuations observées pour la taille peuvent être du même ordre de grandeur que le terme asymptotique dominant, tandis que les fluctuations pour la longueur de cheminement sont toujours d'un ordre de grandeur inférieur par rapport au terme asymptotique dominant.

8.2.2 Taille et longueur de cheminement (alphabet infini)

Dans de l'alphabet infini, la série des «mesures fondamentales uniformes» de profondeur 1 peuvent amener d'autres singularités. Ainsi les longueurs de cheminement des tries hybrides ne restent pas toujours d'ordre $z \log z$.

Nous considérons trois séries de Dirichlet :

$$L(s) := \sum_{i<j} u_j^\star \, u_i^{\star s-1}, \qquad B(s) := \sum_{i<j} \frac{u_j^\star \, u_i^\star}{(u_i^\star + \cdots + u_j^\star)^{2-s}}, \qquad A(s) = \sum_i u_i^{\star s}.$$

La dernière série est exactement la série de Dirichlet $\Lambda_1^{\langle A\rangle}(1,s)$. Par hypothèse (d_3) de la définition 1, il existe $\gamma < 1$ tel que cette série soit convergente pour $\mathrm{Re}(s) > \gamma$. Puisque les objets spectraux dominants Ψ_s et E_s sont strictement positifs pour s réel, la singularité dominante de $\Lambda^{\langle L\rangle}(F,s)$ est en $s = \max(1, s_L)$ où s_L est la singularité dominante de $L(s)$. De même, la singularité dominante de $\Lambda^{\langle B\rangle}(F,s)$ est située en $s = \max(1, s_B)$ où s_B est la singularité dominante de $B(s)$.

Cette discussion nous permet d'affirmer :

PROPOSITION 8.3. (PARAMÈTRES ADDITIFS, MODÈLE DE POISSON, ALPHABET INFINI).

(i) *L'espérance de la longueur de cheminement des tries-abr est toujours de la forme*

$$\widehat{P}_B(z) = O(z \log z).$$

(ii) *Pour tout réel* $1 < \delta < 2$, *il existe une source dynamique pour laquelle la valeur moyenne de la longueur de cheminement des tries-liste est de la forme*

$$\widehat{P}_L(z) = \Theta(z^\delta).$$

Preuve. (i) D'après une idée de S. Janson [58] : En sommant pour $j > i$ les relations, vraies pour $\mathrm{Re}(s) = \sigma < 1$,

$$\frac{u_j^\star}{(u_i^\star + \cdots + u_j^\star)^{2-\sigma}} \le \int_{u_i^\star + \cdots + u_{j-1}^\star}^{u_i^\star + \cdots + u_j^\star} \frac{dt}{t^{2-\sigma}},$$

on obtient pour $\sigma = \mathrm{Re}(s) < 1$

$$|B(s)| \le \sum_i u_i^\star \int_{u_i^\star}^1 \frac{dt}{t^{2-\sigma}} = \frac{1}{\sigma - 1} \sum_i (u_i^\star - u_i^{\star\sigma}) = \frac{A(1) - A(\sigma)}{\sigma - 1}.$$

La quantité $B(s)$ est bien définie pour $\gamma < \mathrm{Re}(s) < 1$, et puisque $A'(1)$ existe, $B(s)$ est aussi bien définie sur la droite $\mathrm{Re}(s) = 1$. Finalement, l'inégalité $|B(s)| \le B(1)$ pour $\mathrm{Re}(s) > 1$ entraîne que $B(s)$ est bien définie pour $\mathrm{Re}(s) > \gamma$. Donc la singularité dominante s_B est strictement inférieure à 1 et l'ordre du terme dominant pour la longueur de cheminement du trie-abr reste en $O(z \log z)$.

(ii) On a la relation $s_B \le s_L \le 2$. En effet, l'inégalité $u_i^\star \le (u_i^\star + \cdots + u_j^\star)$ implique que $B(s) \le L(s)$ pour s réel, $s < 2$.
Si $u_i^\star = \Theta(i^{-\alpha})$ avec $1 < \alpha < 2$, alors $s_L = 2/\alpha$. Dans ce cas, la série $L(s)$ a son terme général en $\Theta(i^{-\rho})$ avec $\rho = \alpha(s-1) + \alpha - 1$.

□

Une telle différence dans les ordres de grandeur des comportements asymptotiques des tries-tableau, trie-liste, tries-abr apparaît par exemple pour une source sans mémoire d'alphabet infini avec les des probabilités $p_k^{(\alpha)}$ d'émettre le k^{e} symbole

$$p_k^{(\alpha)} = \zeta(\alpha)^{-1} k^{-\alpha}, \qquad \alpha > 1,$$

où $\zeta(s)$ est la fonction zeta de Riemann, et α est un paramètre de la distribution. Cette distribution possède une queue de distribution «douce» (le cas $\alpha = 2$ est relativement similaire à l'exemple de la source en fraction continue qui sera étudié plus en détail au chapitre 9).

En conclusion, les tries-ABR assurent contrairement à leurs homologues en liste que le surcoût apporté au niveau de la longueur de cheminement par l'hybridation est du même ordre de grandeur que celui de la longueur de cheminement dans un trie abstrait. L'utilisation de listes peut rendre ce surcoût dominant en ordre de grandeur !

8.3 Dépoissonisation de Dirichlet

La transformée de Mellin est idéale pour traiter le cas de sommes harmoniques. La somme relative à la longueur de cheminement dans les tries-tableau

$$F(x) = \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}} \left(1 - e^{-x u_{\boldsymbol{w}}}\right), \tag{8.6}$$

est un exemple type. Par dépoissonisation algébrique (6.7), (6.8), le modèle de Bernoulli conduit à des sommes dont l'exemple correspondant à (8.6) est

$$G(x) = \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}} \left(1 - (1 - u_{\boldsymbol{w}})^x\right) \tag{8.7}$$

(avec $x = n-1$; voir le théorème 6.15). La détermination du comportement asymptotique de (8.7) est alors plus complexe que dans le cas de (8.6).

L'approche développée ici s'appuie sur l'approximation directe des singularités d'une série de Dirichlet (celle associée au modèle de Bernoulli) par les singularités d'une série plus simple (celle correspondant au modèle de Poisson). Cette approche, appelée *dépoissonisation de Dirichlet*, est brièvement introduite et utilisée pour l'analyse de la recherche multidimensionnelle, mais sans grande justification dans [40]. Le résultat suivant synthétise le principe de la dépoissonisation de Dirichlet.

PROPOSITION 8.4 (DÉPOISSONISATION DE DIRICHLET). *Soit $u_{\boldsymbol{w}} \to 0$ avec $u_{\boldsymbol{w}}$ positif. On définit deux séries de Dirichlet*

$$\Omega(s) = \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}}^s, \qquad \widetilde{\Omega}(s) = \sum_{\boldsymbol{w}\in\mathcal{M}^*} \left(\log \frac{1}{1 - u_{\boldsymbol{w}}}\right)^s.$$

Supposons que $\Omega(s)$ ait un domaine de convergence non vide, admette un prolongement méromorphe au plan complexe $\mathbb{C}$, et soit de croissance polynomiale dans toute bande verticale finie dans le sens faible. Alors les mêmes propriétés sont également vérifiées par $\widetilde{\Omega}(s)$. De plus, les singularités de Ω et celles de $\widetilde{\Omega}$ sont reliées par

$$\mathrm{Sing}(\widetilde{\Omega}) \subseteq \{s - k \mid s \in \mathrm{Sing}(\Omega) \text{ et } k \in \mathbb{N}\},$$

et les expressions singulières de $\Omega(s)$ et $\widetilde{\Omega}(s)$ sont reliées par

$$\widetilde{\Omega}(s) \asymp \Omega(s) + c_1(s)\Omega(s+1) + c_2(s)\Omega(s+2) + \cdots, \tag{8.8}$$

où

$$c_j(s) = [x^j] \exp\left[s \log(\frac{1}{x} \log \frac{1}{1-x})\right]. \tag{8.9}$$

(Le terme «faible» fait à nouveau référence au fait que la croissance doit être contrôlée seulement sur certaines parallèles à l'axe des réels et s'en éloignant vers $\pm i\infty$.)

Preuve. Nous allons justifier la succession de transformations formelles,

$$\begin{aligned}\widetilde{\Omega}(s) &\approx \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}}^s \exp\left(s\log\left(1+\tfrac{1}{2}u_{\boldsymbol{w}}+\tfrac{1}{3}u_{\boldsymbol{w}}^2+\cdots\right)\right)\\ &\approx \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}}^s \exp\left(s\left(\tfrac{1}{2}u_{\boldsymbol{w}}+\tfrac{5}{24}u_{\boldsymbol{w}}^2+\cdots\right)\right)\\ &\approx \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}}^s \left(1+\tfrac{1}{2}su_{\boldsymbol{w}}+(\tfrac{5}{24}s+\tfrac{18}{s^2})u_{\boldsymbol{w}}^2+(\tfrac{1}{8}s+\tfrac{5}{48}s^2+\tfrac{1}{48}s^3)u_{\boldsymbol{w}}^3+\cdots\right)\\ &\approx \Omega(s)+c_1(s)\Omega(s+1)+c_2(s)\Omega(s+2)+\cdots.\end{aligned}$$

Soit σ_0 l'abscisse de convergence de $\Omega(s)$. On fixe également un entier m qui permettra de contrôler l'ordre de grandeur du développement.

Prolongement analytique. Dans un premier temps, nous cherchons à transformer $\Omega(s)$ de façon à assurer l'existence du prolongement analytique, mais nous ne nous inquiéterons pas d'uniformité par rapport à la variable s. Supposons que $\mathrm{Re}(s) > \sigma_0$. D'abord, si $u_{\boldsymbol{w}} \to 0$, on écrit le développement de Taylor

$$\log\left(\frac{1}{u}\log(1-u)^{-1}\right) = D(u) + O(u^{m+1}) \qquad (u\to 0),$$

Pour un polynôme explicite D de degré m. On a donc

$$\widetilde{\Omega}(s) = \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}}^s [u_{\boldsymbol{w}}\log\frac{1}{1-u_{\boldsymbol{w}}}]^s = \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}}^s \exp(sD(u_{\boldsymbol{w}}))\cdot\exp(O(u_{\boldsymbol{w}}^{m+1})).$$

Mais, on l'équivalent suivant

$$\exp(O(u^{m+1})) = 1 + O(u^{m+1}) \qquad (u\to 0),$$

permet de montrer que

$$\Delta(s) := \widetilde{\Omega}(s) - \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}}^s \exp(sD(u_{\boldsymbol{w}})) \tag{8.10}$$

possède un terme général qui décroît comme $O(u_{\boldsymbol{w}}^{\sigma+m+1})$, où $\sigma = \mathrm{Re}(s)$. La fonction $\Delta(s)$ est en particulier analytique pour $\mathrm{Re}(s) \geq \sigma_0 - m$.

À présent, examinons la somme

$$\Omega_1(s) = \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}}^s \exp(sD(u_{\boldsymbol{w}})),$$

pour laquelle on a

$$\exp(sD(u)) = 1 + c_1(s)u + c_2(s)u^2 + \cdots + c_m(s)u^m + O(u^{m+1}),$$

où les c_j sont obtenues grâce à l'équation (8.9). En effet, pour tout s, la quantité $sD(u)$ reste dans un voisinage de 0 et on peut considérer le développement $\exp(v) = 1 + v + \cdots + O(v^{m+1})$. Ainsi, la différence

$$\Delta_1(s) = \Omega_1(s) - (\Omega(s) + c_1(s)\Omega(s+1) + \cdots + c_m(s)\Omega(s+m)) \tag{8.11}$$

possède un terme général qui décroît comme $u^{\sigma+m+1}$. Elle est donc analytique sur $\mathrm{Re}(s) \geq \sigma_0 - m$. Les équations (8.10) et (8.11) montrent finalement que

$$\widetilde{\Omega}(s) = \Delta(s) + \Delta_1(s) + (\Omega(s) + c_1(s)\Omega(s+1) + \cdots + c_m(s)\Omega(s+m))$$

définie au départ pour $\mathrm{Re}(s) > \sigma_0$ est méromorphe pour $\mathrm{Re}(s) \geq \sigma_0 - m$.

Maintenant, en faisant varier m, le caractère méromorphe de $\widetilde{\Omega}(s)$ sur le plan complexe est prouvé. De plus, les singularités de $\widetilde{\Omega}$ sont celles de Ω «décalées» de $0, 1, 2, \ldots$, et l'expression singulière (8.8) est justifiée.

Croissance. Il reste à examiner la croissance de $\widetilde{\Omega}(s)$ lorsque $\mathrm{Im}(s) \to \pm\infty$. Revenons sur la preuve précédente en prenant davantage en compte l'uniformité par rapport à s. Nous supposons que $\mathrm{Re}(s) < \sigma_0 + 1$, puisque c'est le prolongement à gauche qui nous intéresse. L'idée principale réside dans la relation

$$\exp(sO(u^{m+1})) - 1 = \begin{cases} O(1) = O(|s|u^{m+1}) & \text{si } |s|u^{m+1} > 1 \\ O(|s|u^{m+1}) & \text{si } |s|u^{m+1} \leq 1. \end{cases} \tag{8.12}$$

La première ligne se justifie par le fait que u est réel et que $\mathrm{Re}(s)$ majoré. Nous avons alors une borne $O(1)$ à laquelle on substitue la majoration moins précise $O(|s|u^{m+1})$. La seconde ligne dérive du développement usuel de la fonction $e^x - 1$ en 0. En conséquence, la fonction $\Delta(s)$ définie dans l'équation (8.10) est $O(|s|)$ pour $\mathrm{Re}(s) \geq \sigma_0 - m$.

De façon similaire à (8.12), on obtient une version uniforme de (8.10) en séparant les deux cas $|s|u > 1$ et $|s|u \leq 1$,

$$\exp(sD(u)) = 1 + c_1(s)u + c_2(s)u^2 + \cdots + c_m(s)u^m + O(|s|^{m+1}u^{m+1}).$$

Il en résulte que $\Delta_1(s)$ est, pour $\mathrm{Re}(s) \geq \sigma_0 - m$, de croissance au plus $O(|s|^{m+1})$. Ainsi, $\widetilde{\Omega}(s)$ est une somme de fonctions de la forme $c_j(s)\Lambda(s+j)$ et de fonctions analytiques $\Delta(s) + \Delta_1(s)$ qui sont $O(|s|^{m+1})$. La fonction $\widetilde{\Omega}(s)$ est donc de croissance polynomiale faible pour $\mathrm{Re}(s) \geq \sigma_0 - m$. Le résultat est démontré puisque m peut être choisi arbitrairement. □

L'argument s'adapte à des séries de Dirichlet plus générales

$$\Omega(s) = \sum_{\boldsymbol{w} \in \mathcal{M}^*} \lambda_{\boldsymbol{w}} u_{\boldsymbol{w}}^s, \qquad \widetilde{\Omega}(s) = \sum_{\boldsymbol{w} \in \mathcal{M}^*} \lambda_{\boldsymbol{w}} \left(\log \frac{1}{1 - u_{\boldsymbol{w}}} \right)^s,$$

avec $\lambda_{\boldsymbol{w}} > 0$, et la propriété établie dans la proposition 8.4 reste vraie.

8.4 Taille et la longueur de cheminement (Bernoulli)

Le principe de dépoissonisation de Dirichlet montrent que les nouveaux pôles induits sont à une distance au moins un des pôles dans le modèle de Poisson. Par conséquent, les développements asymptotiques du théorème 8.2 restent valides.

THÉORÈME 8.5. (COMPORTEMENT ASYMPTOTIQUE DANS LE MODÈLE DE BERNOULLI : ALPHABET FINI). *Dans le modèle de Bernoulli de paramètre n et pour une source avec un alphabet fini, les équivalents en $O(n)$ pour la taille moyenne des tries et $O(n \log n)$ pour les diverses longueurs de cheminement sont toujours vérifiés. Les expressions données dans le théorème 8.2 restent valides si l'on remplace le paramètre du modèle de Poisson z par le paramètre n du modèle de Bernoulli.*

Il est également possible d'étendre la proposition 8.3 dans le modèle de Bernoulli.

8.5 Analyse asymptotique de la hauteur (Poisson)

Enfin, nous considérons le dernier paramètre, la hauteur. Nous commençons par nous ramener à une somme harmonique à partir de laquelle nous pourrons facilement analyser le comportement asymptotique dans le modèle de Poisson. Puis nous appliquerons la méthode de col pour obtenir le résultat dans le modèle de Bernoulli.

8.5.1 Approximation double exponentielle

L'espérance de la hauteur dans le modèle de Poisson de paramètre z, une source S, et une distribution initiale F, s'exprime sous la forme de la somme infinie (voir théorème 3) :

$$\mathrm{E}[h; \mathcal{P}_z, \mathsf{S}, F] = \sum_{k=0}^{\infty} (1 - \pi_k(z)), \quad \text{avec } \pi_k(z) = \prod_{\boldsymbol{w} \in \mathcal{M}^k} (1 + z u_{\boldsymbol{w}}) e^{-z u_{\boldsymbol{w}}} \tag{8.13}$$

L'analyse de Mellin ne permet pas de traiter directement une telle somme. Mais moyennant une approximation, on peut se ramener une somme harmonique. En effet, la propriété de quasi-puissance de la proposition 7.14 fournit une constante ρ définie par

$$\rho := \lim_{k \to \infty} \frac{\Lambda_k^{\langle A \rangle}(F, 2)}{\lambda(2)^k}. \tag{8.14}$$

Le résultat suivant donne d'abord une approximation de la distribution $\pi_k(z)$ qui est la probabilité dans le modèle $(\mathcal{P}_z, \mathsf{S}, F)$ que le trie construit soit de hauteur au plus k.

PROPOSITION 8.6. *La distribution de la hauteur du trie dans le modèle de Poisson admet une approximation double exponentielle : les deux suites de fonctions*

$$\pi_k(z) = \prod_{\boldsymbol{w} \in \mathcal{M}^k} (1 + z u_{\boldsymbol{w}})\, e^{-z u_{\boldsymbol{w}}}, \qquad \widehat{\pi}_k(z) := \exp\left(-\rho \frac{z^2}{2} \lambda(2)^k\right),$$

satisfont pour tout $z > 0$

$$\sum_{k \geq 0} |\pi_k(z) - \widehat{\pi}_k(z)| = o(1).$$

Preuve. Nous utilisons le fait que $\log \pi_k(z)$ est une somme harmonique avec une série de Dirichlet associée $\Lambda_k^{\langle A \rangle}(F, s)$ et procédons en deux étapes.

Première étape. D'abord, nous comparons $\pi_k(z)$ avec $\widetilde{\pi}_k(z)$ où

$$\widetilde{\pi}_k(z) = \prod_{\boldsymbol{w} \in \mathcal{M}^k} \exp\left(-\frac{z^2 u_{\boldsymbol{w}}^2}{2}\right) = \exp\left(-\frac{z^2}{2} \sum_{\boldsymbol{w} \in \mathcal{M}^k} u_{\boldsymbol{w}}^2\right);$$

On rappelle que $\lambda(s)$ est la valeur propre dominante (pour s réel) commune à tous les opérateurs de Ruelle introduits dans cette thèse. Soit un nombre d dans l'intervalle

$$\left]\frac{3}{|\log \lambda(3)|}, \frac{2}{|\log \lambda(2)|}\right[$$

et posons

$$\kappa(z) := \lfloor d \log z \rfloor.$$

Pour l'analyse de la hauteur, la propriété de log-concavité de la fonction $\lambda(s)$ intervient. En effet l'existence de d est justifiée par l'inégalité stricte

$$\frac{\log \lambda(2)}{\log \lambda(3)} < \frac{2}{3},$$

qui est un cas particulier de la proposition 7.21.

Il existe des nombres positifs ϵ, ϵ' et $\varrho \in [0,1[$ tels que les trois propriétés suivantes soient vérifiées :

$$(C_1):\ z^2\lambda(2)^{\kappa(z)} \geq z^{\epsilon} \quad (C_2):\ z^3\lambda(3)^{\kappa(z)} \leq z^{-\epsilon'}, \quad (C_3):\ (\forall h,\, |h| \geq \kappa(z))\ z u_{\boldsymbol{w}} \leq \varrho. \tag{8.15}$$

La somme dans (8.13) est coupée en deux parties à l'indice $\kappa(z)$.

(i) *Les indices du début.* Nous considérons la partie de la somme pour les indices $k \leq \kappa(z)$. L'inégalité (vraie pour $x \geq 0$)

$$\exp\left(-\frac{x^2}{2}\right) \leq (1+x)e^{-x} \leq \exp\left(-\frac{x^2}{2(1+x)}\right)$$

implique que

$$|\widetilde{\pi}_k(z) - \pi_k(z)| \leq \exp\left(-\frac{z^2}{2}\sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}}^2\right) - \exp\left(-\frac{z^2}{2}\sum_{\boldsymbol{w}\in\mathcal{M}^k} \frac{u_{\boldsymbol{w}}^2}{1+zu_{\boldsymbol{w}}}\right).$$

Définissant la suite

$$B_k = \frac{1}{2}\sum_{\boldsymbol{w}\in\mathcal{M}^k} \frac{u_{\boldsymbol{w}}^2}{1+zu_{\boldsymbol{w}}},$$

et utilisant l'inégalité

$$\sum_{i\in I} \frac{a_i^2}{1+a_i} \leq \frac{s^2}{1+s}$$

valable pour toute suite positive $(a_i)_{i\in I}$ dont la somme est s, nous en déduisons que la suite B_k est décroissante. Avec le théorème des accroissements finis, on obtient l'inégalité

$$|\widetilde{\pi}_k(z) - \pi_k(z)| \leq z^3 \exp\left(-z^2\, B_{\kappa(z)}\right).$$

Dès lors, la propriété (C_3) de l'indice $\kappa(z)$, la propriété de quasi-puissance, et enfin la propriété (C_1) de l'indice $\kappa(z)$ montrent l'inégalité

$$\sum_{k\leq\kappa(z)} \widetilde{\pi}_k(z) - \pi_k(z) \leq \kappa(z)\, z^3\, \exp\left(-dz^{\epsilon}\right) = o(1). \tag{8.16}$$

(ii) *Les indices de la fin.* La deuxième partie de la somme, relative à $k > \kappa(z)$ est également $o(1)$. En effet, l'inégalité

$$\pi_k(z) - \widetilde{\pi}_k(z) \leq \log \pi_k(z) - \log \widetilde{\pi}_k(z)$$

et les relations

$$\log \pi_k(z) \leq \sum_{\boldsymbol{w}\in\mathcal{M}^k} \left(-\frac{(zu_{\boldsymbol{w}})^2}{2} + \frac{(zu_{\boldsymbol{w}})^3}{3}\right), \qquad \log \widetilde{\pi}_k(z) = -\sum_{\boldsymbol{w}\in\mathcal{M}^k} \frac{(zu_{\boldsymbol{w}})^2}{2},$$

entraînent que

$$|\pi_k(z) - \widetilde{\pi}_k(z)| \leq z^3 \sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}}^3.$$

Une fois encore, la propriété de quasi-puissance en $s = 3$ et la propriété (C_2) de l'indice $\kappa(z)$ donnent les inégalités suivantes

$$\sum_{k\geq\kappa(z)} |\pi_k(z) - \widetilde{\pi}_k(z)| \leq d_0 z^3 \sum_{k\geq\kappa(z)} \lambda(3)^k = d_0 z^3 \frac{\lambda(3)^{\kappa(z)}}{1-\lambda(3)} \leq z^{-\epsilon'} = o(1). \tag{8.17}$$

Deuxième étape. La distribution $\widetilde{\pi}_k(z)$ dans le modèle de Poisson admet une bonne approximation de la forme $\widehat{\pi}_k(z) = \exp\left(-\rho\frac{z^2}{2}\lambda(2)^k\right)$. La propriété de quasi-puissance entraîne une fois encore l'existence de deux constantes d_1 et d_2 vérifiant les deux inégalités

$$\left|\sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}}^2 - \rho\lambda(2)^k\right| \leq d_1\,\nu^k, \qquad \sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}}^2 \geq 2d_2\,\lambda(2)^k,$$

où ν est strictement comprise entre la valeur propre dominante $\lambda(2)$ et le module $\mu(2)$ de la valeur propre sous-dominante de l'opérateur $\mathfrak{G}_2$. Ainsi, on aboutit avec l'inégalité des accroissements finis à

$$\left|\sum_{k\geq 0} \widetilde{\pi}_k(z) - \widehat{\pi}_k(z)\right| \leq \sum_{k\geq 0} |\widetilde{\pi}_k(z) - \widehat{\pi}_k(z)| \leq d_1 \frac{z^2}{2} \sum_{k\geq 0} \nu^k \exp\left(-d_2\,\frac{z^2}{2}\,\lambda(2)^k\right).$$

Par des moyens élémentaires ou en utilisant la transformée de Mellin, on obtient alors que

$$z^2 \sum_{k\geq 0} \nu^k \exp\left(-d_2\frac{z^2}{2}\lambda(2)^k\right) = o(1). \tag{8.18}$$

Finalement, les équations (8.16), (8.17) et (8.18) entraînent le résultat. □

8.5.2 Distribution de la hauteur dans le modèle de Poisson

Les équations (8.16), (8.17) et (8.18) fournissent la distribution asymptotique de la hauteur $\widehat{h}_z$ dans le modèle $(\mathcal{P}_z, \mathsf{S}, F)$, puisque ces équations garantissent

$$\lim_{z\to\infty} \sup_{k\geq 0} \left|\Pr\left[\widehat{h}_z < k\right] - \exp\left(-\rho\frac{z^2}{2}\lambda(2)^k\right)\right| = 0.$$

La somme harmonique, approximation de la hauteur moyenne dans le modèle de Poisson, s'écrit

$$D(z) = \sum_{k=0}^{\infty} \left(1 - e^{-\rho\frac{z^2}{2}\lambda(2)^k}\right),$$

et a pour transformée de Mellin

$$D^*(s) = -\tfrac{1}{2}\,(\rho/2)^{-s/2}\,\frac{\Gamma(s/2)}{1-\lambda(2)^{-s/2}}.$$

La bande fondamentale est $\langle -2, 0\rangle$, et l'expression singulière en $s=0$ est

$$D^*(s) \asymp -\frac{2}{\log\lambda(2)}\frac{1}{s^2} + \left[\frac{\gamma+\log\rho-\log 2}{\log\lambda(2)} - \frac{1}{2}\right]\frac{1}{s} \qquad (s=0).$$

Par ailleurs, l'existence de pôles régulièrement espacés sur la droite $\mathrm{Re}(s)=0$ correspondant aux solutions de

$$\lambda(2)^{s/2} = 1,$$

entraînent l'existence de fluctuations périodiques. Ces fluctuations sont de faible amplitude du fait de la décroissance exponentielle de la fonction $\Gamma(s)$ lorsque l'on s'éloigne de l'axe réel (voir l'équation (8.4)).

THÉORÈME 8.7 (HAUTEUR ASYMPTOTIQUE DANS LE MODÈLE DE POISSON). *Dans le modèle de Poisson de paramètre z relatif à une source S avec une distribution initiale F, la valeur moyenne de la hauteur est*

$$\widehat{H}(z) = \frac{2}{|\log\lambda(2)|}\log z + Q_F(\log z) - \left(\frac{\gamma+\log\rho-\log 2}{\log\lambda(2)} - \frac{1}{2}\right) + o(1),$$

où ρ est une constante positive dépendant de la source dynamique probabilisée (S, F) et $Q_F(u)$ est une fonction périodique d'amplitude très petite.
De plus, la distribution asymptotique de la hauteur est de type double exponentiel,

$$\lim_{z\to\infty}\sup_{k\geq 0}\left|\Pr\{\widehat{h}_z < k\} - \exp\left[-\rho z^2\lambda(2)^k\right]\right| = 0.$$

8.6 Hauteur dans le modèle de Bernoulli

Nous rappelons les notations du théorème 6.17. La quantité $\pi_{k,n}$ désigne la probabilité qu'un trie construit sur un n-uplet de mots aléatoires produits par une source dynamique probabilisée ait une hauteur au plus k. La fonction génératrice exponentielle $\Pi_k(z)$ des $\pi_{k,n}$

$$\Pi_k(z) := \sum_n \pi_{k,n}\frac{z^n}{n!}, \quad \text{s'écrit } \Pi_k(z) := \prod_{\boldsymbol{w}\in\mathcal{M}^k}(1+zu_{\boldsymbol{w}}).$$

Dans le cas d'une source sans mémoire, on peut poursuivre l'analyse en utilisant la méthode du point de col [33, 44] et elle devient même complètement élémentaire dans le cas de tries binaires symétriques (cf. [71]). Nous proposons de suivre précisément la même approche afin d'analyser la hauteur dans le cas d'une source dynamique générale. Le cadre général de la «dépoissonisation par la méthode de col» est présenté dans un article de Jacquet et Szpankowski [57].
Formellement, l'idée est la suivante : On part de la formule intégrale de Cauchy

$$\pi_{k,n} = \frac{n!}{2i\pi}\int_\gamma \Pi_k(z)\,\frac{dz}{z^{n+1}},$$

où γ est une courbe simple orientée positivement entourant l'origine. Puisque, pour tout $x \in [0, +\infty[$ et tout entier m fixé, on a

$$\log(1+x) = \frac{x}{1} - \frac{x^2}{2} + \frac{x^3}{3} - \cdots + (-1)^{m-1}\frac{x^{m-1}}{m-1} + O(x^m),$$

la transformation «exp-log» appliquée sous le signe intégral donne

$$\begin{aligned}\Pi_k(z) &= \exp \sum_{\boldsymbol{w}\in\mathcal{M}^k} \log(1+zu_{\boldsymbol{w}}) \\ &= \exp\left(z\sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}} - \frac{z^2}{2}\sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}}^2 + \frac{z^3}{3}\sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}}^3 - \cdots + O(z^m \sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}}^m)\right),\end{aligned} \tag{8.19}$$

où les sommes sont prises sur les mots de longueur k. En ne retenant que les deux premiers termes dans (8.19), on peut espérer, au moins pour des valeurs convenables de k, l'approximation suivante

$$\pi_{k,n} \approx \frac{n!}{2i\pi}\int_\gamma \exp\left(-\frac{z^2}{2}\sum u_{\boldsymbol{w}}^2\right)\frac{e^z\,dz}{z^{n+1}}. \tag{8.20}$$

Cela peut être vu comme une perturbation de la formule intégrale de Cauchy appliquée à e^z. La formule de Cauchy donnant le coefficient d'une série génératrice exponentielle $\widehat{g}(z)$ est

$$g_n = n!\,[z^n]\,\widehat{g}(z) = n!\frac{1}{2i\pi}\int_\gamma \frac{\widehat{g}(z)\,dz}{z^{n+1}}.$$

Il est bien connu qu'une intégrale comme celle de l'équation (8.20) peut être estimée grâce à la méthode de col [20, 43] qui consiste en deux étapes

(i) intégrer sur le cercle $|z| = n$ (une approximation du point de col) ;

(ii) observer que la contribution est concentrée sur un petit secteur $ne^{i\theta}$ avec $|\theta| \le \theta_0$ et $\theta_0 = n^{-1/2+\alpha}$ pour tout α tel que $0 < \alpha < \frac{1}{6}$ (cette condition permet d'assurer simultanément que $n\theta_0^2 \to +\infty$ tandis que $n\theta_0^3 \to 0$) ;

(iii) enfin, réduire l'intégrale asymptotiquement à une intégrale Gaussienne complète qui peut être évaluée.

Dans le cas de e^z, c'est bien sûr une des façons standard d'obtenir la formule de Stirling pour $1/n!$. Si nous considérons le terme $\exp(-z^2\sum_{|\boldsymbol{w}|=k} u_{\boldsymbol{w}}^2)$ comme presque constant sur la petite partie du contour d'intégration qui importe, nous sommes conduits aux approximations

$$\begin{aligned}\pi_{k,n} &\approx \frac{n!}{2i\pi}\exp\left(-\frac{n^2}{2}\sum u_{\boldsymbol{w}}^2\right)\int_{\gamma_0}\frac{e^z\,dz}{z^{n+1}} \\ &\approx \exp\left(-\frac{n^2}{2}\sum u_{\boldsymbol{w}}^2\right) &(8.21)\\ &\approx \exp\left(-\rho\frac{n^2}{2}\lambda(2)^k\right), &(8.22)\end{aligned}$$

étant donné l'intégrale de col de e^z/z^{n+1}, de la formule de Stirling pour (8.21), et les résultats relatifs aux intervalles fondamentaux pour (8.22). L'estimation (8.22) n'est rien d'autre que l'approximation double exponentielle de la distribution, déjà rencontrée dans le modèle de Poisson. La constante ρ déjà définie dans (8.14) intervient également.

THÉORÈME 8.8 (HAUTEUR ASYMPTOTIQUE DANS LE MODÈLE DE BERNOULLI). *Dans le modèle de Bernoulli de paramètre n relatif à une source S avec une distribution initiale F, l'approximation uniforme double exponentielle reste valide en remplaçant le paramètre z du modèle de Poisson par le paramètre n du modèle de Bernoulli*

$$\Pr\{H_n \leq k\} = \exp(-\rho \frac{n^2}{2} \lambda(2)^k)(1 + o(1))$$

où ρ est une constante positive dépendant de la source dynamique probabilisée (S, F).

Preuve. Nous expliquons plus rigoureusement les arguments exposés plus haut. Soit α une petite constante avec $0 < \alpha < \frac{1}{6}$. On pose

$$\theta_0 = n^{-1/2+\alpha}, \quad \gamma = \{ne^{i\theta} \mid |\theta| \leq \pi\}, \quad \gamma_0 = \{ne^{i\theta} \mid |\theta| \leq \theta_0\}.$$

On doit distinguer deux domaines de variation de k avec une frontière $\kappa_1(n)$ qui est définie par

$$\kappa_1(n) = \lfloor d \log n \rfloor,$$

pour une constante d choisie afin de satisfaire les contraintes suivantes qui ne sont que des raffinements des contraintes rencontrées lors de l'étude dans le modèle de Poisson (voir (8.15)).

Il existe $\epsilon > 0$ pour lequel les conditions suivantes sont vérifiées :
(C_1) $d < 2/|\log \lambda(2)|$, ce qui assure que pour $k = \kappa_1(n)$, le terme $n^2 \lambda(2)^k$ est exactement $\Theta(n^\epsilon)$.
(C_2) $d > 2/|\log \nu|$, où ν est strictement compris entre la valeur propre dominante $\lambda(2)$ et le module de la valeur propre sous-dominante de l'opérateur $\mathfrak{G}_2$. Cela rend l'approximation $n^2 \sum u_{\boldsymbol{w}}^2 \sim \rho n^2 \lambda(2)^k$ vraie avec un terme d'erreur absolue $O(n^{-\epsilon})$ dès que $k \geq \kappa_1(n)$ (équation (7.51)).
(C_3) $d > 3/|\log \lambda(3)|$, si bien que pour $k \geq \kappa_1(n)$, les termes $n^3 \lambda(3)^k$ sont $O(n^{-\epsilon})$ et peuvent être négligés ;
(C_4) $d > (3/2+\alpha)/|\log \lambda(2)|$, ce qui assure que les termes $n^2 \lambda(2)^k \theta_0$ sont $O(n^{-\epsilon})$ pour $k \geq \kappa_1(n)$. C'est une contrainte technique nécessaire pour valider l'estimation obtenue par perturbation du point de col.

Les contraintes (C_1) et (C_3) sont compatibles étant donné la propriété de log-concavité de $\lambda(\sigma)$. Les contraintes (C_1) et (C_2) est vérifiée puisque la valeur propre sous-dominante est de module inférieur (strictement) à $\lambda(2)$. Les contraintes (C_2) et (C_4) sont automatiquement vérifiée avec $\alpha \in]0, 1/2[$. En conséquence, la constante d peut être choisie dans l'intervalle

$$\left] \max\left(\frac{3}{|\log \lambda(3)|}, \frac{2}{|\log \nu|}, \frac{3/2+\alpha}{|\log \lambda(2)|} \right), \frac{2}{|\log \lambda(2)|} \right[.$$

(i) *Indices du début.* Soit $z = re^{i\theta}$, on calcule facilement

$$\log|1 + re^{i\theta}| == \tfrac{1}{2} \log(1 + 2r\cos\theta + r^2) = r\cos\theta - \tfrac{r^2}{2} \cos 2\theta + O(r^3).$$

On obtient donc la suite d'inéquations sur le cercle $z = ne^{i\theta}$ et à l'indice $k = \kappa_1(n)$

$$\begin{aligned}\log|\Pi_k(z)| &= \sum_{|\boldsymbol{w}|=k} \log|1 + zp_{\boldsymbol{w}}| \\ &= n\cos\theta - \frac{n^2}{2} cos2\theta \sum_{|\boldsymbol{w}|=k} p_{\boldsymbol{w}}^2 + O\left(n^3 \sum_{|\boldsymbol{w}|=k} p_{\boldsymbol{w}}^3\right) \\ &= n\cos\theta - \rho\frac{n^2}{2} cos2\theta\lambda(2)^2 + O\left(n^2\nu^k\right) + O\left(n^3\lambda(3)^k\right) \\ &= n\left(\cos\theta - \rho\frac{n}{2}\cos 2\theta\lambda(2)^k\right) + O\left(n^{-\epsilon}\right). \end{aligned} \tag{8.23}$$

On a par une étude élémentaire que la fonction $g(\theta) = cos\theta - c\cos 2\theta$ avec $0 < c < \frac{1}{2}$ admet un maximum sur $]-\pi, \pi[$ atteint en $\theta = 0$. On obtient donc (puisque $n\lambda(2)^k = o(1)$) à l'indice $k = \kappa_1(n)$ que pour n suffisamment grand

$$\log|\Pi_k(z)| \leq n\cos\theta - c_0 n^{\epsilon_0},$$

pour des constantes $c_0 > 0$ et $\epsilon_0 > 0$. Ainsi, des bornes triviales appliquées à l'intégrale de Cauchy

$$\pi_{k,n} \leq n!\frac{1}{R^n}\max_{|z|=R}|\Pi_k(z)|$$

entraînent pour une approximation du point de col $R = n$

$$\pi_{\kappa_1(n),n} \leq \frac{n!e^n}{n^n}\cdot e^{-c_0 n^{\epsilon_0}}, \tag{8.24}$$

ce qui correspond à une queue de distribution à gauche, $k \leq \kappa_1(n)$, globalement exponentiellement petite.

(ii) *Les indices du domaine central et de la fin.* On découpe le contour γ en deux :

1. Pour $k \geq \kappa_1(n)$ et sur la partie $\gamma \setminus \gamma_0$ du contour, on a encore l'équivalent (8.23). Le maximum est atteint en $\theta = \theta_0$. La contribution de l'intégrale dans la région non-centrale $\gamma\setminus\gamma_0$ satisfait donc

$$\left|\frac{n!}{2i\pi}\int_{\gamma\setminus\gamma_0}\Pi_k(z)\frac{dz}{z^{n+1}}\right| \leq \frac{n!}{n^n}\sup_{\gamma\setminus\gamma_0}|\Pi_k(z)| = O\left(\exp\left[-\left(n\theta_0^2/4 + \rho\frac{n^2}{2}\lambda(2)^k\right)\right]\right). \tag{8.25}$$

   (On remplace $\theta_0^2/2$ par $\theta_0^2/4$ afin d'absorber les termes de la forme $\sqrt{n}$ qui proviennent de la formule de Stirling). D'après (8.25), la contribution sur $\gamma\setminus\gamma_0$ est ainsi exponentiellement petite (puisque $n\theta_0^2 = n^{2\alpha}$).

2. Dans le domaine γ_0, on écrit la quantité $\Pi_k(z)$ sous la forme

$$\Pi_k(z) = \exp\left(z - \rho\frac{z^2}{2}\lambda(2)^k\right)\exp(A(z) + B(z)),$$

   où $A(z)$ et $B(z)$ sont définies sur γ_0 et satisfont $|A(z)| = O(\nu^k|z|^2)$ et $|B(z)| = O(\lambda(3)^k|z|^3)$. On peut donc écrire avec $z = ne^{i\theta}$ et grâce aux conditions (C_2), (C_3) et

(C_4)

$$\Pi_k(z) = e^z \exp(-\rho n^2 \lambda(2)^k) \exp\left(-\rho \frac{n^2}{2}(e^{2i\theta} - 1) + A(z) + B(z)\right)$$
$$= e^z \exp(-\rho \frac{n^2}{2} \lambda(2)^k)(1 + \epsilon(z)), \tag{8.26}$$

où $|\epsilon(z)| = O(|z|^{-\beta})$ (avec $\beta > 0$).
Puisque la formule intégrale de Cauchy appliquée à e^z donne $1/n!$, étant donné la borne (8.25) sur $\gamma \setminus \gamma_0$ et l'approximation uniforme (8.26) sur γ_0, on conclut que pour $k \geq \kappa_1(n)$,

$$\pi_{k,n} = \exp[-\rho \frac{n^2}{2} \lambda(2)^k] \left(1 + O(\frac{1}{n^{\alpha_3}})\right). \tag{8.27}$$

Les équations (8.24) et (8.27) caractérisent complètement le domaine complet $k \geq 1$. On établit ainsi une approximation du même type que celle établie dans le modèle de Poisson, ce qui complète la preuve. □

Aparté méthodologique. La preuve donnée ci-dessus revient essentiellement à «relever» les estimations établies pour de grandes valeurs *réelles* du paramètre z au cas où z décrit (tout ou partie) du cercle complexe $z = ne^{i\theta}$. C'est cohérent avec les principes généraux de dépoissonisation par méthode de col [33, 57]. Il est également à noter, en concordance avec [33], que la limite double exponentielle pour la distribution de la hauteur n'est pas une probabilité de distribution limite dans le sens classique du terme. Posant

$$\kappa_0(n) := \left\lfloor \frac{\log(\rho n^2/2)}{|\log \lambda(2)|} \right\rfloor$$

et ne tenant pas compte des termes d'erreur, on a

$$\pi_{\kappa_0(n)+j,n} \approx e^{-\beta(n)\lambda(2)^j} \qquad \text{où} \qquad \beta(n) = \left(\frac{1}{\lambda(2)}\right)^{\{\log(\rho n^2/2)/|\log \lambda(2)|\}}, \tag{8.28}$$

où $\{u\}$ désigne la partie fractionnaire. En d'autres termes, la distribution est un échantillonage discret de la fonction double exponentielle, comme elle est exprimée dans (8.27). De manière équivalente, il existe une famille (8.28) de distributions (gouvernée par le terme borné $\beta(n)$) qui varie graduellement et périodiquement lorsque n croit comme une puissance de $1/\lambda(2)$. Un tel phénomène périodique exprimé à l'aide d'un échantillonage discret d'une distribution de valeur extrême revient souvent pour les tries ; voir par exemple [62] pour une discussion plus détaillée.

Chapitre 9

Sources particulières & expérimentations

Sommaire

Nous finissons cette étude par un chapitre présentant des résultats pour des sources particulières et quelques expérimentations. Les sources sans mémoire et les chaînes de Markov sont ici vues du point de vue des sources dynamiques. Il est intéressant de voir quels sont les opérateurs impliqués, quelles formes ont les valeurs propres et les fonctions propres. La source en fraction continue fournit une connexion intéressante avec des problèmes mathématiques profonds (en l'occurrence les zéros non triviaux de la fonction zeta de Riemann).

La structure très efficace de TST fait l'objet d'expérimentations sur un corpus (*Moby Dick, d'H. Melville*). Les résultats cadrent remarquablement bien avec l'analyse théorique. La structure hybride de TST a également permis l'implantation du correcteur orthographique `epelle` [17]. Les différents choix techniques sont expliqués. En particulier, la forme d'un TST peut être optimisée pour accélérer la recherche. D'autre part l'algorithme de compression de l'arbre (consistant à mettre en commun certaines portions de l'arbre) est présenté.

9.1 Applications

Nous développons brièvement trois cas particuliers : les sources sans mémoire, les chaînes de Markov et la source en fraction continue. Les résultats sont systématiquement donnés dans le modèle plus naturel de Bernoulli de paramètre n.

9.1.1 Sources sans mémoire

Les sources sans mémoire sont des sources sur un alphabet fini ou infini $\mathcal{M}$, où le symbole m est émis de façon indépendante avec la probabilité p_m. L'opérateur usuel de Ruelle associé au système est alors

$$\mathcal{G}_s[f](z) := \sum_{m\in\mathcal{M}} p_m^s\, f(q_m + p_m z), \qquad \text{avec} \quad q_m := \sum_{i<m} p_i.$$

La fonction propre dominante est la même pour chaque opérateur composante $\mathcal{G}_{s,h}$. Cette fonction est une fonction constante par rapport à la variable s pour $\mathrm{Re}(s) > \sigma$. Cette fonction constante est également la fonction propre dominante de $\mathcal{G}_s$, pour toutes les valeurs de s, et $\lambda(s) = \sum_{m\in\mathcal{M}} p_m^s$ est la valeur propre associée. Le projecteur dominant $e_s[f]$ est la forme intégrale $\int_0^1 f(t)dt$. Plus généralement, le spectre de $\mathcal{G}_s$ est

$$\mathrm{Sp}\,\mathcal{G}_s = \{\lambda_\ell(s) := \sum_{m\in\mathcal{M}} p_m^{s+\ell},\ \ell \geq 0\},$$

et, donc, le déterminant de Fredholm est

$$\mathcal{F}(s,u) = \prod_{\ell\geq 0} [1 - u \sum_{m\in\mathcal{M}} p_m^{\ell+s}].$$

Le vecteur propre relatif à la ℓ^{e} valeur propre $\lambda_\ell(s)$ est un polynôme de degré ℓ. Pour les sources sans mémoire symétriques, cette ℓ^{e} valeur propre est indépendante de s et dans le cas d'un alphabet binaire, la famille de fonctions propres coïncide exactement avec les polynômes de Bernoulli [11], définis par

$$B_\ell(z) := \ell!\, [t^\ell]\, \frac{te^{zt}}{e^t - 1}.$$

Les résultats décrits dans le corollaire suivant sont classiques et concernent les arbres digitaux construits sur un alphabet fini $\mathcal{M} = \{1, 2, \ldots, r\}$ où le symbole i est émis avec une probabilité p_i. On a[1]

COROLLAIRE 9.1. *Soit une source* B *sans mémoire avec un alphabet fini de cardinalité* r *et des probabilités* $\{p_i\}_{i=1}^r$. *L'entropie et la probabilité de coïncidence sont*

$$h(\mathsf{B}) = -\sum_i p_i \log p_i, \qquad c(\mathsf{B}) = \sum_i p_i^2.$$

(i) *Pour un trie standard, la taille et la longueur de cheminement ont pour espérances*

$$S(n) \approx \frac{1}{h(\mathsf{B})} n, \qquad P(n) \sim \frac{n \log n}{h(\mathsf{B})}.$$

(ii) *La hauteur d'un trie standard suit asymptotiquement une loi de distribution double exponentielle d'espérance*

$$H(n) \sim \frac{2}{|\log c(\mathsf{B})|}\, \log n.$$

[1]Nous utilisons la notation '$\approx$' pour 'approximativement égal', (par opposition à '$\sim$' réservé pour le plus précis «asymptotiquement équivalent»), i.e., à quelques fluctuations près provenant de pôles non réels.

(iii) *Les longueurs de cheminement des tries hybrides satisfont*

$$P_L(n) \sim \frac{K_L(\mathsf{B})}{h(\mathsf{B})} n \log n, \qquad P_B(n) \sim \frac{K_L(\mathsf{B})}{h(\mathsf{B})} n \log n,$$

où

$$K_L(\mathsf{B}) = \sum_{i=1}^{r} (i-1) p_i, \qquad K_B(\mathsf{B}) = 2 \sum_{i<j} \frac{p_i p_j}{p_i + \cdots + p_j}.$$

Les expressions pour la hauteur et la longueur de cheminement étendent les analyses de [80, 95, 96] au cas d'une densité analytique. L'analyse des tries hybrides construits sur de telles sources a été faite dans l'article [16]. Devroye [21] a considéré l'effet d'une densité non nécessairement analytique avec une source sans mémoire non-biaisée (ou symétrique, i.e. avec $p_1 = p_2 = \frac{1}{2}$). Devroye a ainsi montré que, soit $H(n) \sim 2 \log n$, soit $H(n) = +\infty$, selon que la quantité $\int f^2$ (où f est la densité) existe ou non, et que $P(n) \sim n \log n$ dès que f est de carré intégrable.

Si la source est périodique, les fluctuations périodiques sont du même ordre de grandeur que le terme dominant pour la taille. Les sources sans mémoire suivantes sont périodiques :

$$(1/2, 1/4, 1/4), \qquad (p, p, p^2) \text{ avec } p = (1/2)(\sqrt{2} - 1), \qquad (p_m)_{m \geq 1} \text{ avec } p_m = (1/2)^m.$$

Le cas où le déterminant de Fredholm est pseudo-périodique de période t, i.e., il existe $a \neq 2k\pi$ tel que

$$\mathcal{F}(s + it, u) = \det(I - u\mathcal{G}_{s+it}) = \mathcal{F}(s, e^{ia} u)$$

est également intéressant. Une source sans mémoire est pseudo-périodique si et seulement si il existe deux nombres réels a et b tels que b n'appartienne pas au groupe cyclique $\langle a \rangle$ engendré par a et que tous les nombres p_m/b appartiennent au groupe cyclique $\langle a \rangle$. Un exemple d'une telle situation est $(2/5, 2/5, 1/5)$. Comme l'ont noté Pollicott [81] et Fayolle et al. [27], il y a une accumulation de points s pour lesquels $\lambda(s) = 1$ à gauche de la droite $\mathrm{Re}(s) = 1$. L'argument pour la proposition 8.4 adaptant l'inversion de la transformée de Mellin à ce cas montre directement que la contribution totale de ces pôles est en $o(n)$ pour la taille et $o(n \log n)$ pour la longueur de cheminement.

9.1.2 Chaînes de Markov

Nous considérons maintenant le cas particulier des chaînes de Markov. Ici, l'alphabet $\mathcal{M}$ est de cardinal fini r, et la matrice Π_s dont le terme général est $p^s_{i|j}$ joue un rôle extrêmement important. Pour $s = 1$, c'est la matrice de transition de la chaîne de Markov.
Le spectre de l'opérateur-matrice $\mathcal{G}_s$ est exactement la réunion des spectres des matrices $\Pi_{s+\ell}$, avec $\ell \geq 0$. On a donc

$$\mathrm{Sp}\, \mathcal{G}_s = \bigcup_{\ell \geq 0} \mathrm{Sp}\, \Pi_{s+\ell} \qquad \mathcal{F}(s, u) = \prod_{\ell \geq 0} \det(I - u\, \Pi_{s+\ell}).$$

Si les valeurs propres de la matrice Π_s sont notées $\lambda^{(i)}(s)$ pour $1 \leq i \leq r$, alors

$$\mathrm{Sp}\, \mathcal{G}_s = \{\lambda^{(i)}(s + \ell)\, | 1 \leq i \leq r,\ \ell \geq 0\},$$

et toutes les composantes du vecteur propre correspondant à la valeur propre $\lambda^{(i)}(s + \ell)$ sont des polynômes de degré ℓ. La valeur propre dominante de l'opérateur $\mathcal{G}_s$ est exactement la valeur propre dominante de la matrice Π_s, et les fonctions propres associées ont toutes leurs composantes constantes. Enfin, la fonction propre Ψ_1 n'est autre que le vecteur de probabilités stationnaire de la chaîne de Markov.

COROLLAIRE 9.2. *Soit une chaîne de Markov* M *avec des probabilités de transition* $\{p_{i|j}\}_{1\le i,j\le r}$. *L'entropie et la probabilité de coïncidence sont*

$$h(\mathsf{M}) = -\sum_k \pi_k \sum_j p_{j|k} \log p_{j|k}, \qquad c(\mathsf{M}) = \lambda(2)$$

où π_k *est la* k^e *composante du vecteur stationnaire de probabilités de la chaîne de Markov, et* $\lambda(2)$ *est la valeur propre du carré terme à terme de la matrice* $(p^2_{i|j})$ *(i.e. de la matrice obtenue en élevant chaque terme au carré).*
(i) *La taille du trie et sa longueur de cheminement ont des espérances de la forme*

$$S(n) \approx \frac{1}{h(\mathsf{M})} n, \qquad P(n) \sim \frac{1}{h(\mathsf{M})} n \, \log n.$$

(ii) *La hauteur d'un trie admet une loi de distribution asymptotique de type double exponentielle avec pour espérance*

$$H(n) \sim \frac{2}{|\log \lambda(2)|} \, \log n.$$

(iii) *Les valeurs moyennes des longueurs de cheminement pour les tries hybrides satisfont*

$$P_L(n) \sim \frac{K_L(\mathsf{M})}{h(\mathsf{M})} n \log n, \qquad P_B(n) \sim \frac{K_B(\mathsf{M})}{h(\mathsf{M})} n \log n,$$

avec

$$K_L(\mathsf{M}) = \sum_k \pi_k \sum_i (i-1) p_{i\,|\,k}, \quad K_B(\mathsf{M}) = 2 \sum_k \pi_k \sum_{i<j} \frac{p_{i|k} p_{j|k}}{p_{i|k} + \cdots + p_{j|k}}.$$

Comme mentionné dans l'introduction, ce résultat se rapporte à la référence [54]. Les calculs de constantes sont expliqués un peu plus en détail dans l'annexe B.

9.1.3 La source en fraction continue.

Des résultats préliminaires sur la source en fraction continue se trouvent dans [45], mais se limitent à une densité uniforme. L'opérateur de Ruelle dans ce cas est appelé opérateur de Ruelle-Mayer,

$$\mathcal{G}_s[f](z) := \sum_{m\ge 1} \frac{1}{(m+z)^{2s}} \, f(\frac{1}{m+z}),$$

et la convergence est assurée pour tout nombre complexe s satisfaisant $\mathrm{Re}(s) > 1/2$.
L'entropie de la source est connue sous le nom de constante de Lévy, et joue un rôle clé dans la théorie métrique des fractions continues et l'analyse de l'algorithme d'Euclide. La probabilité de coïncidence $\lambda(2)$ correspondant à l'opérateur de Ruelle-Mayer est une constante parfois appelée «constante de Vallée», et intervient dans les généralisations en dimension 2 de l'algorithme d'Euclide [19, 45, 99]. (Pour ce versant des choses, voir également [45] et la bibliographie attenante.)

COROLLAIRE 9.3. *Soit une source en fraction continue* CF *pour une fonction de densité initiale analytique. L'entropie et la probabilité de coïncidence sont*

$$h(\mathsf{CF}) = -\lambda'(1) = \frac{\pi^2}{6 \log 2} \qquad c(\mathsf{CF}) = \lambda(2) \doteq 0.19945\,88183\,43767.$$

(i) *Les valeurs moyennes de la taille et de la longueur de cheminement satisfont*

$$S(n) = \frac{6\log 2}{\pi^2} n + o(n), \qquad P_A(n) = \frac{6\log 2}{\pi^2}\, n\log n + O(n).$$

(ii) *La hauteur obéit asymptotiquement à une loi de distribution de type double exponentielle avec pour espérance*

$$H(n) = \frac{2}{|\log \lambda(2)|} \log n + O(1).$$

(iii) *Les valeurs moyennes des longueurs de cheminement des tries hybrides satisfont*

$$P_L(n) \sim \frac{3\log 2}{2\pi^2}\, n \log^2 n$$

$$P_B(n) \sim Cn\log n \quad \textit{avec} \quad C := \frac{3}{\pi^2}\left[1 + 4\sum_{k\geq 2} \frac{1}{k^2-1}\log\frac{k+1}{2}\right] \doteq 0.81125\,48533\,15373.$$

Le calcul des constantes est détaillé dans l'annexe A.

L'exemple de la source en fraction continue est intéressant par bien des aspects. Tout d'abord, comme mentionné (iii) au-dessus, les longueurs de cheminement $P_L(n)$ pour les tries-liste et $P_B(n)$ pour les tries-abr s'avèrent ne pas être asymptotiquement du même ordre de grandeur lorsque n tend vers $+\infty$. (On pouvait cependant s'y attendre puisque la source en fraction continue ressemble à une source sans mémoire de type Zipf généralisé, avec des probabilités $\zeta(\alpha)\frac{1}{i^\alpha}$, avec $\alpha = 2$; voir la section 8.2.2). Ensuite, les problèmes liés à la périodicité (ou l'apériodicité) sont particulièrement fascinants.

La source en fraction continue est apériodique, et les pôles de $(I - \mathcal{G}_s)^{-1}$ interviennent dans des problèmes mathématiques très profonds : ils incluent les zéros non triviaux de la fonction zeta de Riemann. Les autres valeurs de s pour lesquelles $\mathcal{G}_s$ admet une valeur propre égale à 1 sont reliées aux opérateurs hyperboliques de Laplace. Quoiqu'il en soit, ces dernières valeurs n'apparaissent pas comme pôles de la série de Dirichlet $\Lambda^{\langle A\rangle}(Id, s)$. En effet, dans le demi-plan $\mathrm{Re}(s) > 0$, la série de Dirichlet $\Lambda^{\langle A\rangle}(\mathrm{Id}, s)$ peut être représentée par (voir l'annexe A)

$$\Lambda(s) \equiv \Lambda^{\langle A\rangle}(Id, s) = 2\frac{\zeta(2s-1)}{\zeta(2s)}\,\frac{2^{1-s}-1}{1-s} + \frac{R(s)}{\zeta(2s)}$$

où $R(s)$ est analytique pour $\mathrm{Re}(s) > 0$. Ainsi, si la densité initiale est uniforme, le développement asymptotique de $S(n)$ ou $P_A(n)$ donné dans le théorème 8.5 ne dépend que des zéros non triviaux de la fonction zeta de Riemann. Pour une densité non uniforme, la même situation se produit, car les vecteurs propres des opérateurs hyperboliques de Laplace satisfont $\int_0^1 f(x)dx = 0$ [72]. Ainsi, la nature de l'asymptotique du second ordre des paramètres additifs est reliée à l'hypothèse de Riemann.

COROLLAIRE 9.4. *Les fluctuations de la valeur moyenne de la longueur de cheminement et de la taille des tries standard avec une source en fraction continue sont reliées à l'hypothèse de Riemann. Par exemple pour la taille, on a*

$$S(n) = \frac{6\log 2}{\pi^2} n + \Phi(n) + O(1),$$

où $\Phi(n)$ est une somme indicée par l'ensemble Z des zéros de la fonction zeta de Riemann dans la bande «critique» $0 < \mathrm{Re}(s) < 1$,

$$\Phi(n) = \sum_{\chi \in Z} \mathrm{Res}\left(\Lambda(-s)(s+1)\Gamma(s)n^{-s}\right)_{s=-\chi/2}. \tag{9.1}$$

En particulier, sous l'hypothèse de Riemann (R.H.), on a $\Phi(n) = O(n^{1/4+\epsilon})$ pour tout $\epsilon > 0$. De plus, sans tenir compte de R.H., les deux implications suivantes sont valides.
(i) *Si $\sigma_0 = \sup\{\mathrm{Re}(\xi),\ \xi \in Z\}$, alors, pour tout $\epsilon > 0$, on a*

$$\lim_{n\to\infty} \frac{\Phi(n)}{n^{\sigma_0/2+\epsilon}} = 0.$$

(ii) *Réciproquement, si σ_1 est le supremum de tous les σ tels que $\Phi(n)/n^{\sigma/2}$ est non borné, alors $\zeta(s)$ possède au moins un zéro dans $\mathrm{Re}(s) \geq \sigma_1$.*

Preuve. L'équation (9.1) s'ensuit, du moins formellement, du calcul de résidu sur un grand rectangle $(-11/10 \pm iT, 1/10 \pm iT)$. L'équation (9.1) est justifiée par le fait que $\zeta(s)$ croît assez vite le long de lignes horizontales qui traversent la bande «critique» à haute altitude, évitant ainsi les zéros de la bande critique. De plus, $1/\zeta(s)$ est de croissance polynomiale faible dans une bande qui contient la bande critique. Cette dernière assertion est valide du fait de propriétés connues de la fonction zeta de Riemann ; les détails sont omis et sont complètement similaires à ceux développés dans la discussion de la formule de Ramanujan dans le livre de Titchmarsh [97, sections 9.7 et 9.8] ; voir également [24].
Une fois l'équation (9.1) établie, les autres implications s'ensuivent. □

Aspects numériques des fluctuations. Nous offrons une discussion succincte sur la fonction fluctuante $\Phi(n)$. Nous savons, d'après la discussion précédente, que la fonction $\Lambda(s)$ est de croissance polynomiale faible. Soit $\Phi^{[T_0]}(n)$ la somme tronquée issue de (9.1) restreinte aux zéros ξ de la fonction zeta tels que $|\mathrm{Im}(\xi)| \leq T_0$. La différence $|\Phi(n) - \Phi^{[T_0]}(n)|$ est le produit d'un polynôme en T_0 et d'une quantité qui est approximativement $e^{-\pi T_0/2} n^{1/2}$, étant donné le caractère de décroissance exponentielle de la fonction Gamma en $\pm i\infty$. Dans cette discussion, nous utilisons $\doteq$ dans le sens informel de 'grossièrement égal'.
Prenons par exemple $T_0 = 5 \times 10^8$ correspondant aux premiers 1.5×10^9 zéros de la fonction zeta, d'après les calculs de van de Lune, te Riele et Winter en 1986. Ces zéros satisfont tous l'hypothèse de Riemann. Oubliant les facteurs polynomiaux, une borne supérieure sur le terme d'erreur lié à la troncature, lorsque les termes correspondant aux grands zéros de Z sont négligés, est alors au pire

$$|\Phi(n) - \Phi^{[T_0]}(n)| \leq E_1(n) \quad \text{où} \quad E_1(n) \doteq \left(10^{-6\times 10^9} \cdot n\right)^{1/2}.$$

C'est ainsi complètement négligeable pour toutes les valeurs de n inférieures à un seuil d'à peu près $10^{6\times 10^9}$.
Pour toutes les applications «pratiques» l'hypothèse de Riemann peut être considérée comme «vraie». Si nous nous concentrons sur les quelques premiers termes de (9.1), deux choses interviennent : la forte décroissance de la fonction Gamma et le fait que les premiers zéros de la fonction zeta possèdent une partie imaginaire assez grande, étant situés en $\xi_1, \xi_2 = \frac{1}{2} \pm 14.134i$. Supposons dans un souci d'illustration que le résidu de $\Lambda(-s)$ en ξ_1, ξ_2 soit de module au plus 50, la contribution de $\Phi(n)$ –celle qui dominerait numériquement de toute façon– est majorée par $(8 \times 10^{-7}) \cdot n^{1/4}$. Les zéros suivants sont encore plus loin, là où la fonction Gamma est encore plus petite. Nous pouvons donc nous attendre au fait que

$$|\Phi(n)| \leq 10^{-6} \cdot n^{1/4} \quad \text{pour} \quad n \leq 10^{6\times 10^9}.$$

Cette quantité devient 1 seulement pour des valeurs de n autour de 10^{24}.
En résumé, l'hypothèse de Riemann, qu'elle soit vraie ou non, n'affecte pas vraiment la nature des fluctuations. De plus, $\Phi(n)$, même si elle oscille de $-\infty$ à $+\infty$, ne sera pas détectable à moins que n devienne très grand ($n > 10^{24}$). Les fluctuations seront en particulier complètement absorbées par le terme constant du développement asymptotique de $S(n)$. Ce fait est intéressant puisqu'il fournit une instance naturelle du «phénomène de Dumont» (découvert par Dumont en 1980 et prouvé rigoureusement par Delange). Ce phénomène surprenant réside dans le fait que, empiriquement, pour des valeurs de x très proches de 1, la convergence numérique

$$\sum_{n=1}^{\infty} \mu(n) x^n \xrightarrow{?} \frac{1}{\zeta(0)} = -2, \qquad (x \to 1^-),$$

où $\mu(n)$ est la fonction de Moebius, paraît valide alors qu'en réalité la partie gauche oscille finalement de façon non bornée pour $n \to +\infty$ (voir [46] pour une discussion plus approfondie).

9.2 Expérimentation

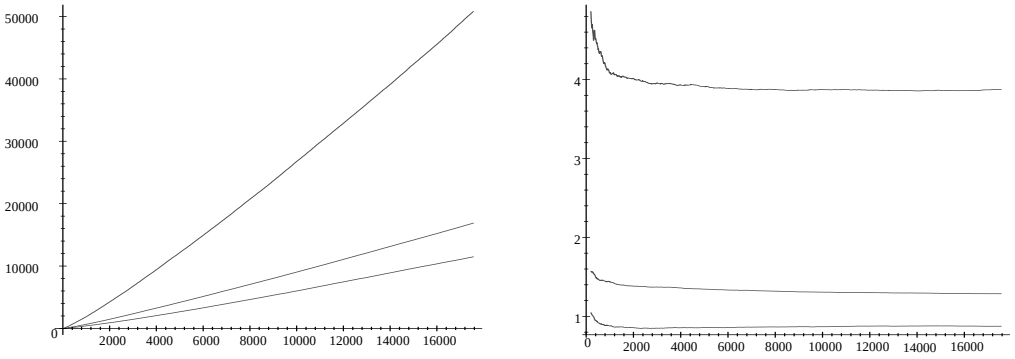
9.2.1 Expérimentation sur Mobydick(H. Melville)

Données expérimentales. Nous avons voulu appuyer l'étude théorique effectuée dans les sections précédentes par une campagne de simulation sur un corpus textuel conséquent. À l'origine, nous espérions que les structures de données se comporteraient *qualitativement* de manière compatible avec les modèles théoriques. Une adéquation *quantitative* nous semblait hors de portée étant donné la simplicité de nos modèles (mots infinis, chaînes de Markov du premier ordre, régime asymptotique non établi). La situation réelle est en fait plutôt meilleure qu'attendue.

Comme corpus de référence, nous avons choisi le roman d'Herman Melville *Moby Dick* dont le texte complet est disponible sur Internet. Le livre entier contient $1\,295\,000$ caractères et $217\,000$ mots —suite de lettres alphabétiques contiguës— dont les longueurs s'échelonnent de 1 (le mot 'a') à 20 ('uninterpenetratingly'). Certains expériences ont été conduites, comme séparer le texte en deux, filtrer les mots courts (de longueur inférieure à 5 par exemple), etc. Il y a $17\,448$ mots distincts dans le roman. On construit facilement la source sans mémoire et la chaîne de Markov pour ce corpus particulier. Les valeurs théoriques estimées pour l'entropie $h(\mathsf{S})$ et les constantes $K_N(\mathsf{S})$ du théorème 8.2 sont calculées d'après les sections précédentes (sections 9.1.1 et 9.1.2)

$$\begin{aligned} h(\mathsf{B}) &= 2.896, \quad h(\mathsf{M}) = 2.271; \\ K_L(\mathsf{B}) &= 10.649, \quad K_L(\mathsf{M}) = 10.447; \\ K_B(\mathsf{B}) &= 3.298, \quad K_B(\mathsf{M}) = 2.295. \end{aligned}$$

Dans ces formules, la source sans mémoire et la chaîne de Markov sont notées respectivement B et M. (Comme prévu, la chaîne de Markov possède une entropie plus petite et on trouve des



(i) Longueur de cheminement (ii) Mise en évidence de la constante

FIG. 9.1 – La longueur de cheminement pour les tries hybrides est en $O(n \log n)$. La figure de droite permet de mettre en évidence la constante en divisant par $n \log n$. Les courbes de haut en bas représentent les contributions dues aux listes, aux ABR et la longueur de cheminement du trie standard.

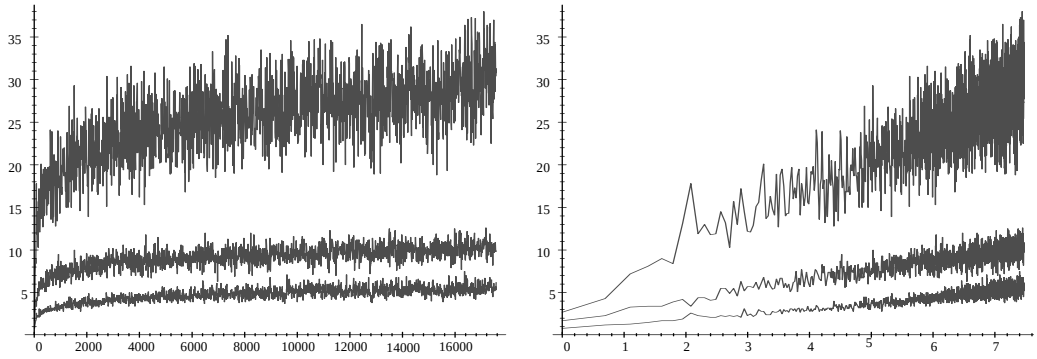
valeurs d'entropie en accord avec celles obtenues sur d'autres textes [104].) Les espérances des longueurs de cheminement P_A, P_L, P_B dans les tries standard, listes et ABR contenant n mots sont donc

source sans mémoire B :	$0.345\, n \log n,$	$3.677\, n \log n,$	$1.138\, n \log n$
chaîne de Markov M :	$0.440\, n \log n,$	$4.600\, n \log n,$	$1.010\, n \log n.$

Résultats et courbes. Plusieurs statistiques sont intéressantes concernant les tries hybrides. On peut tracer les courbes correspondant aux longueurs de cheminement (figure 9.1). Sur chaque dessin on superpose trois types de longueur de cheminement : la longueur de cheminement du trie proprement dit (courbe la plus basse), la longueur de cheminement provenant de l'hybridation avec les ABR (courbe médiane) et les listes ordonnées (courbe haute). La «véritable» longueur de cheminement d'un trie hybride est donc la somme de la longueur de cheminement du trie abstrait et de celle provenant de l'hybridation.
Ces statistiques nous permettent d'apprécier la pertinence du modèle choisi. En particulier, la deuxième courbe confirme le comportement en $O(n \log n)$.

L'évolution des coûts d'insertion (figure 9.2), ou de façon équivalente des coûts de recherche négative, est croissante avec le nombre de mots n (distincts) insérés. Les données numériques sont relativement dispersées. En Fig. 9.2-(i), on représente le coût d'insertion en fonction de n (avec les contributions de haut en bas : des liens de type «liste», des liens de types «ABR», des liens de type «trie»). En Fig. 9.2-(ii), les mêmes données sont représentées en échelle logarithmique. Fort logiquement, le coût d'une insertion est du même ordre que la profondeur moyenne d'une feuille c.-à-d. $O(\log n)$.

Un aspect intéressant auquel nous allons plus particulièrement nous attacher dans la section suivante se situe dans le fait que les mots d'un texte n'apparaissent pas avec la même fréquence (loi de Zipf par exemple). On peut ainsi tracer la trajectoire de la longueur de cheminement pour un



(i) Coût d'insertion

(ii) Coût d'insertion (échelle logarithmique)

FIG. 9.2 – Évolution du coût d'insertion sur les mots (distincts) du dictionnaire de Moby Dick [de haut en bas : trie-liste, trie-ABR, trie standard].

texte donné (Fig. 9.3). Pour chaque mot du texte (déjà présent ou non dans le trie), on affiche la valeur de la longueur de cheminement. Il s'agit juste d'un changement d'échelle sur l'axe des abscisses.

Nous décrivons une expérience en détail. En prenant la première partie du texte et en retenant les mots de longueur supérieure ou égale à 6, on obtient un ensemble de $7\ 437$ mots. Le nombre de nœuds internes est $8\ 444$. Les longueurs de cheminement du trie standard (P_A), trie-liste (P_L) et trie-ABR (P_B) sont respectivement $P_A = 50\ 894$, $P_L = 178\ 688$, $P_B = 59\ 715$. Les formules empiriques pour les différentes longueurs de cheminement sont

$$\mathrm{E}_n[P_A] \approx 0.9n \log n, \qquad \mathrm{E}_n[P_B] \approx 1.3n \log n, \qquad \mathrm{E}_n[P_L] \approx 3.8n \log n.$$

Les valeurs théoriques calculées sont optimistes et sont grossièrement la moitié de ce à quoi on s'attendait pour la taille (le nombre de nœuds internes) et la longueur de cheminement du trie standard. Une interprétation plausible de ce fait provient du fait que de nombreux mots sont très proches les uns des autres dans un dictionnaire construit à partir de toutes les formes d'un mot (*i.e.*, ici, aboriginal, aboriginally, aboriginalness). Les prédictions pour les tries-liste et les tries-ABR sont très proches de celles qui sont observées.
En résumé, on en arrive à la conclusion :

> *Les* TST *constituent une structure de données efficace du point de vue de la théorie de l'information puisqu'une recherche nécessite typiquement* $\log n$ *comparaisons. Les tries-liste demandent à peu près trois fois autant de comparaisons. Pour un alphabet de taille 26, la structure de* TST *occupe 9 fois moins d'espace mémoire que la version standard (avec tableaux de pointeurs).*

Cela justifie de considérer les TST comme une structure de choix pour traiter de larges volumes de données textuelles. En citant [8], "*Ternary search tries combine the best of two worlds : the low overhead of binary search trees (in terms of space and running time) and the character-based efficiency of tries*".

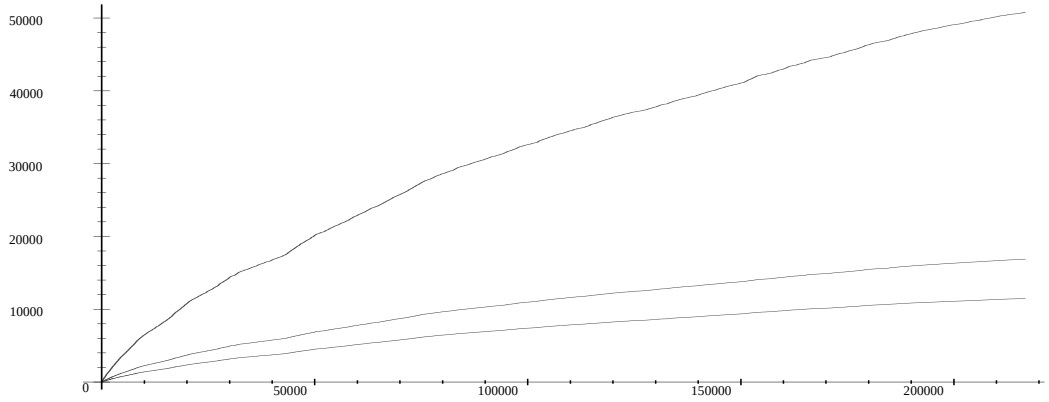


FIG. 9.3 – Trajectoire observée de la longueur de cheminement en insérant les mots du texte un par un (avec les répétitions).

9.2.2 Un exemple de correcteur orthographique : `epelle`

La détection de fautes d'orthographes est un vieux problème en informatique. Elle consiste à vérifier que tous les mots appartiennent bien au dictionnaire. L'algorithme utilisé par le programme `epelle` permet de vérifier avec un taux d'erreur nul [17]. D'autres méthodes existent où l'on autorise un certain taux d'erreur en échange d'une économie en mémoire [7].

Présentation

Le programme `epelle` [105] a été créé par Paul Zimmermann et repose sur la structure d'arbre digital en liste. Fort des résultats de cette thèse sur la suprématie, il était tentant d'introduire la structure de TST pour rendre le programme plus performant !

Le but de ce vérificateur orthographique est de proposer une structure d'arbre digital TST compact afin de représenter un dictionnaire. Le programme originel `epelle` utilisait une méthode de compactification qui évite la duplication de parties de l'arbre. Cela correspond à une situation courante pour un dictionnaire où un grand nombre de terminaisons sont partagées par tout un ensemble de préfixes. C'est particulièrement frappant pour les règles de conjugaisons des verbes du premier groupe par exemple (-e, -es, -e, -ons, -ez, -ent).

Structure de données

Le programme `epelle` utilisait dans la version précédente les tries-liste et chaque nœud possédait 4 champs : un lien de descente dans le trie, un lien vers le nœud frère dans l'arbre, un caractère et enfin un booléen indiquant si le nœud correspond à un mot du dictionnaire.

Le choix pris ici est d'utiliser la structure proposée à la section 5.3.1. On adopte la représentation du langage C selon laquelle la fin des mot est repérée par le caractère `'\0'`. Chaque nœud possède également quatre champs (trois pointeurs et un caractère). Le code ne différencie plus différents types de nœuds et devient plus efficace. L'inconvénient apparaît au niveau de l'espace mémoire nécessaire. La structure utilisée est celle du chapitre 5.

```
typedef struct tnode *Tptr;
typedef struct tnode {
   char splitchar;
   Tptr lokid, eqkid, hikid;
} Tnode;
```

Réorganisation du TST

Par ailleurs, la structure de TST propose l'avantage par rapport à celle du trie-liste d'avoir des possibilités d'optimisation ; celles-ci seront donc exploitées.

Un même TST peut avoir des formes très différentes. À titre d'exemple, la figure 9.4 représente deux TST construits sur le même texte (en l'occurrence le premier chapitre de Moby Dick), selon que les mots sont insérés en ordre lexicographique ou dans l'ordre où les mots apparaissent dans le texte.
Des algorithmes permettent d'optimiser les ABR par rapport à une série de requêtes. Certains de ces algorithmes sont dynamiques et s'adaptent à l'entrée (les splay trees par exemple [93]) mais ne sont pas en général optimaux. D'autres sont plus statiques et nécessitent la connaissance *a priori* des fréquences des symboles stockés dans l'ABR pour équilibrer celui-ci (algorithme de Ku-Tucker). Étant donnée la structure statique du dictionnaire, c'est le deuxième choix qui a été fait dans l'implantation.

On peut optimiser la forme d'un TST par rapport à la longueur de cheminement de l'arbre (somme des longueurs des chemins pour aller de la racine à chacune des feuilles), ou bien plus généralement par rapport à un corpus particulier. La deuxième solution est la plus adaptée pour les données textuelles. En effet, le mot the par exemple n'apparaît qu'une fois dans le dictionnaire alors que ce même mot peut faire l'objet de nombreuses requêtes.

La loi de Zipf est une loi observée empiriquement sur les textes en langue naturelle : les fréquences des mots suivent une loi proportionnelle à $1/n$. Sur l'exemple de Moby Dick, on peut rassembler quelques statistiques. Le premier chapitre de Moby Dick comprend 2242 mots. Le tableau suivant rassemble les 10 premiers mots dans un classement par nombre d'occurrences.

mot	nombres d'occurrences	pourcentage
the	125	5.57%
of	81	3.61%
and	73	3.26%
a	68	3.03%
to	53	2.36%
in	48	2.14%
i	43	1.92%
is	34	1.52%
that	31	1.38%
as	26	1.11%
	582	25.95 %

À la lecture de ce tableau, il paraît avantageux d'optimiser le TST par rapport à cet ensemble de mots puisqu'un mot sur quatre dans le texte en fait partie. Pour donner une idée plus précise de la portion de l'arbre qui «supporte» la plus grosse charge, on dessine les même arbres qu'à la figure 9.4 en ne représentant que les nœuds correspondant à ces dix mots (figure 9.5).

L'algorithme de Ku-Tucker [63] permet d'optimiser la forme d'un ABR par rapport aux requêtes. Le modèle sur les données est celui du multiensemble (vu au chapitre 6). Pour un multi-ensemble $E = \{n_1 \cdot 1, n_2 \cdot 2, \ldots, n_r \cdot r\}$, l'algorithme de Ku-Tucker permet d'obtenir l'ABR sur $\{1, 2, \ldots, r\}$

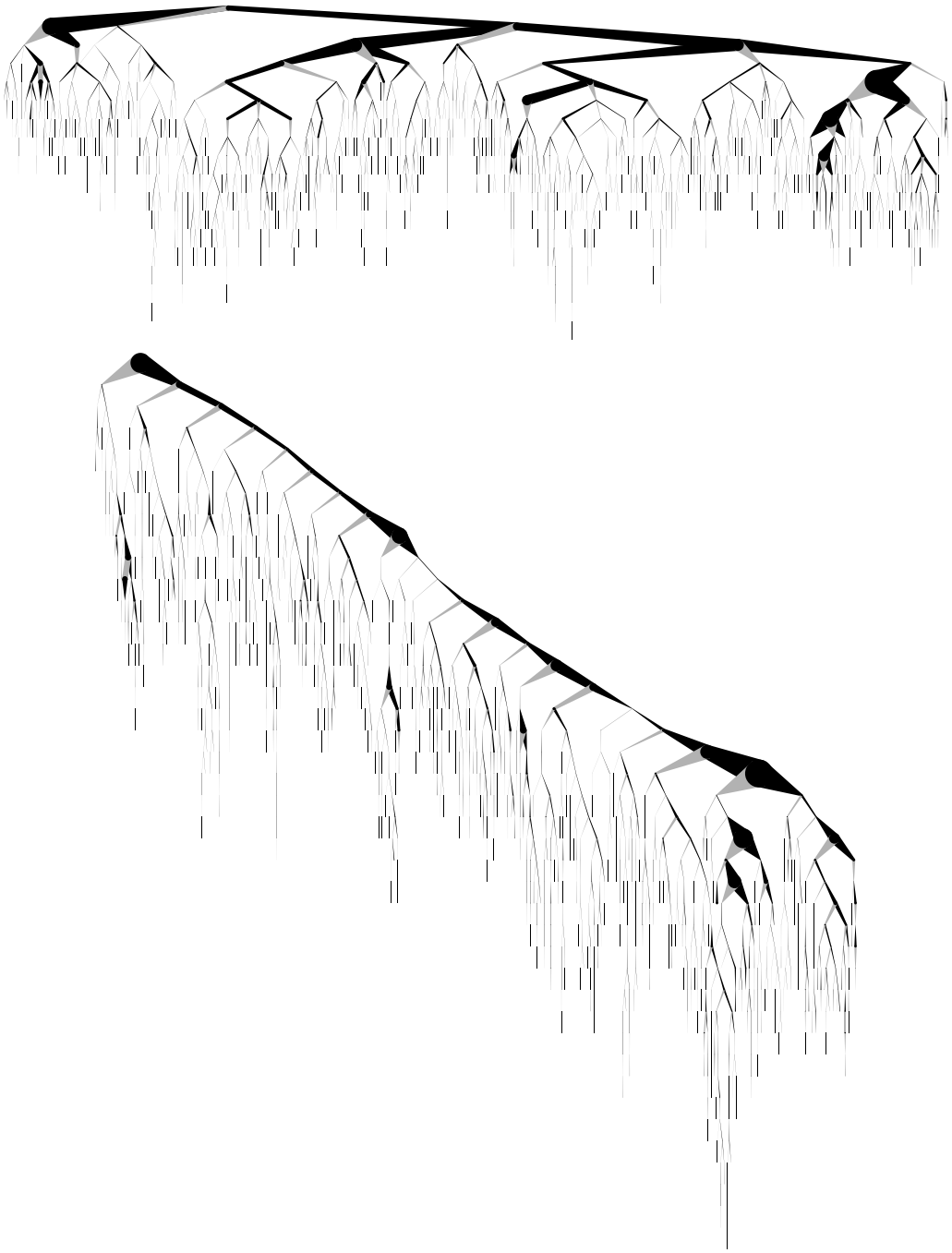


FIG. 9.4 – Deux TST construits sur les mots insérés dans le même ordre que celui de leur apparition dans le texte (à haut) et en ordre trié (en bas). La grosseur d'un nœud est proportionnelle au nombre d'accès pour une recherche de tous les mots du texte (avec les répétitions). Les liens de type «abr» sont en noir tandis que les liens de type «trie» sont en gris.

optimal par rapport au coût de recherche de l'ensemble des clés de E (c.-à-d. n_1 fois la recherche de la clé 1, n_2 fois la recherche de la clé 2, etc.). Cet algorithme de construction est en $O(r^2)$.

Pour construire le TST optimal, on rassemble les statistiques aux nœuds. Ensuite pour chaque structure-nœud, on procède à l'optimisation. Avec un alphabet de taille finie r, puisque le nombre de nœuds internes est de l'ordre de n, le coût de l'optimisation est en moyenne $O(nr^2)$. L'arbre correspondant au premier chapitre de Moby Dick est représenté sur la figure 9.6.

Compression de l'arbre

Afin de limiter l'espace mémoire nécessaire, l'arbre est compressé en mettant en commun les sous arbres communs (ce qui fait penser aux graphes acycliques sur les mots ; DAWG). La comparaison de deux sous-arbres s'effectue en associant à chaque sous-arbre un entier à partir des champs de sa racine. D'où l'algorithme suivant, utilisant une structure `S` globale pour stocker les entiers alloués pour les sous-arbres déjà rencontrés.

```
Algorithme compactifie(t)
  /* renvoie l'entier associé à l'arbre t */
  si t == NULL alors
    renvoyer 0;
  i = compactifie(T->lokid);
  j = compactifie(T->eqkid);
  k = compactifie(T->hikid);

  m = cherche(S, i, j, k, T->splitchar);
  si (m < 0) alors {
    m = creer_etiquette(i, j, k, T->splitchar);
    ajouter(S, m);
  }
  renvoyer m;
```

La fonction `cherche` renvoie l'entier correspondant au sous-arbre attaché au nœud courant s'il est dans `S` (c.-à-d. qu'il a déjà été rencontré) et -1 sinon. La fonction `creer_etiquette` renvoie un entier non alloué (cela correspond au fait de dire que le sous-arbre rencontré est unique jusqu'à présent).

Les TST construits sur des dictionnaires se compressent relativement bien. En effet de nombreux suffixes sont partagés. On va étudier ici deux exemples : le premier correspond au dictionnaire des mots contenu dans Moby Dick (un dictionnaire assez réduit de $17\ 568$ mots), le second est le dictionnaire français fourni avec `epelle` ($260\ 689$ mots). Nous indiquons des taux de compression en nombre de nœuds. Le format choisi ici pour l'écriture sous forme de fichier de l'arbre compressé est de stocker dans une table de taille maximale $2^{18} = 262\ 144$ les nœuds de l'arbre compressé en codant les liens sur 18 bits. Par ailleurs les liens nuls ne sont pas codés. On pourrait très facilement améliorer le codage (ne serait-ce qu'en utilisant un code de Huffman ; cf chapitre 2) mais, bien sûr, le chargement de la table en est ralenti d'autant. Les résultats sont rassemblés dans le tableau suivant (la machine utilisée est un pentium II 266MHz).

fichier	mots	taille fichier	nœuds avant compression	nœuds après compression	taille table	temps création/ optimisation/ compression (en ms)
français	260 689	2 764 Ko	819 710	73 212	354 Ko	830/1 120/1 750
Moby Dick	17 568	144 Ko	65 335	17 960	85 Ko	50/90/180

La dernière colonne de ce tableau indique le temps de création du TST à partir du texte (chargé en mémoire principale), le temps pour optimiser le TST (équilibrer les ABR grâce à l'algorithme de Ku-Tucker) et enfin le temps pris par la compression de l'arbre.

FIG. 9.5 – Figure correspondant à la figure 9.4 où seuls les nœuds correspondant à l'ensemble the, of, and, a, to, in, i, is, that sont représentés.

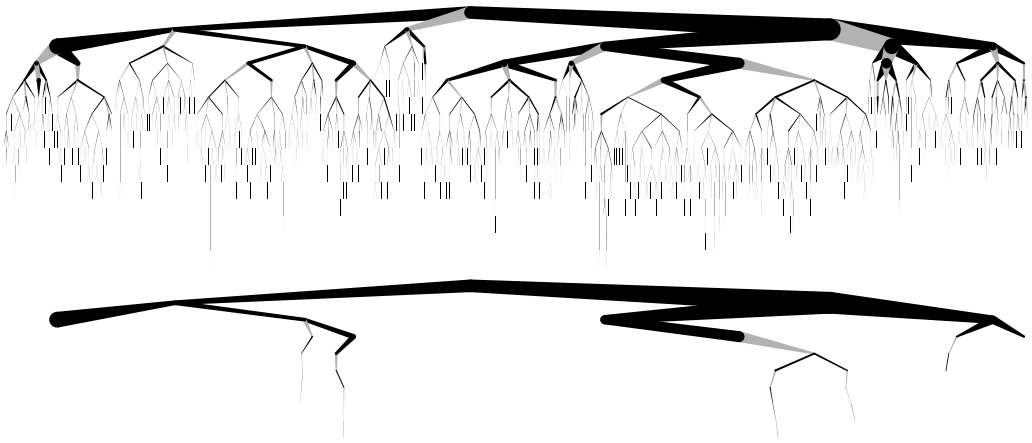


FIG. 9.6 – Le TST correspondant au chapitre 1 de Moby Dick optimisé [en haut : l'arbre entier, en bas : seuls les nœuds correspondant aux mots the, of, and, a, to, in, i, is, that sont représentés].

On connaît assez bien la répartition des motifs dans les arbres binaires de recherche [31]. Dans un ABR de taille n construit par insertions aléatoires, chaque motif apparaît avec une fréquence qui est en moyenne proportionnelle à n. Une telle étude pourrait sans doute être également menée pour les tries standards et permettrait de mieux comprendre les phénomènes de compression observés ici.

Chapitre 10

Conclusion

Cette thèse fournit une double généralisation. D'une part elle fournit un nouveau modèle unificateur englobant un grand nombre de sources classiques. D'autre part, elle s'intéresse à des structures hybrides plus proches de l'implantation.

Les études en moyenne d'arbres digitaux partent généralement d'un modèle probabiliste simple sur les mots. Un premier pas vers une plus grande généralité est de considérer les chaînes de Markov. Nous proposons dans cette thèse d'aller plus loin avec un modèle unificateur, celui des sources dynamiques probabilisées. Ce nouveau modèle englobe un grand nombre de sources classiques : sources sans mémoire, chaînes de Markov ou le système de numération en fraction continue. Ces sources tirent leurs origines de la physique statistique. Empruntant au formalisme de la thermodynamique, on associe à une source un opérateur générateur pourvu de propriétés spectrales dominantes. Des grandeurs caractéristiques de la source comme l'entropie ou la probabilité de coïncidence s'expriment alors directement.

Dans un souci de se rapprocher davantage de l'implantation, cette thèse pose un cadre d'étude des tries hybrides. Autour de la structure d'arbre digital ou de trie (le squelette), on greffe en chaque nœud une structure permettant d'accéder à ses fils. Le TST (*ternary search trie*) résulte d'une telle hybridation avec des arbres binaires de recherche. L'étude théorique confirme le bon comportement de cette structure en pratique. Comme souligné par Bentley et Sedgewick, il s'agit d'une structure de données très efficace pour les données textuelles qui dispose de nombreuses fonctionnalités (par rapport aux tables de hachage notamment).

Cette thèse propose un panel assez complet des techniques et outils utilisés en analyse d'algorithme (séries génératrices, transformée de Mellin, méthodes de poissonisation et de dépoissonisation). De plus l'incursion inédite des systèmes dynamiques dans le champ de l'analyse d'algorithme ouvre des perspectives intéressantes.

Extensions et problèmes ouverts. Des études préliminaires tendent à montrer qu'il serait possible d'utiliser les méthodes de cette thèse et le modèle de source pour analyser les arbres de suffixes, en dépit des difficultés dues aux corrélations intervenant alors ; voir [55] pour une étude dans un cadre classique. La structure de k-d trie (trie à k dimensions) semble également pouvoir tirer profit des travaux présentés ici.

Un autre problème intéressant réside dans la capacité de compression que semblent offrir les tries. L'étude de motifs dans les tries est sans doute accessible par des méthodes symboliques et permettrait de mieux comprendre les phénomènes de compression.

La structure d'arbre digital se marie particulièrement bien avec le nouveau modèle des sources dynamiques. Sans doute cela s'explique-t-il en partie par la correspondance entre les préfixes des mots émis par la source et les intervalles fondamentaux (engendrés par les opérateurs générateurs). Cette notion d'intervalle fondamental, très riche, constitue un des concepts centraux de cette thèse. La correspondance entre préfixes et intervalles fondamentaux fournit également des liens avec la théorie de l'information.

Les sources générales sont d'ores et déjà employées pour analyser d'autres paramètres en moyenne autour d'arbres digitaux comme la taille des arbres PATRICIA [13], la hauteur de pile d'un trie [12] (la taille de la pile nécessaire pour un parcours récursif du trie). Cette notion de source dynamique peut être considérée avec succès pour l'analyse en moyenne de problèmes autour des mots (*pattern matching* à l'aide d'automates par exemple).

La hauteur d'un trie standard est un paramètre bien connu aujourd'hui. Les moments et la distribution sont précisément caractérisés. D'autres problèmes intéressants sont dans le prolongement direct de cette thèse et restent ouverts.

– L'analyse de la *hauteur des tries hybrides* (tries-liste et tries-ABR), où la hauteur est définie comme la longueur de la chaîne de pointeurs maximale reliant la racine à l'une des feuilles, semble requérir des méthodes plus probabilistes que celles exposées dans cette thèse. Il faut en effet arriver à combiner les deux types de structures (trie et ABR) et à déterminer quelle structure influence le plus la hauteur.
– Pour les *paramètres additifs avec un modèle de source général*, nous avons peu d'informations sur la distribution même dans le cas du trie standard. Une approche particulièrement prometteuse est celle utilisée par Jacquet et Szpankowski. La concentration de la distribution est très vraisemblable pour la taille et la longueur de cheminement. Cela revient à dire que l'écart type est d'ordre strictement inférieur à la moyenne. On peut également conjecturer qu'il existe une distribution limite gaussienne pour ces deux paramètres dans la plupart des cas (ce qui est suggéré par les études avec des modèles de sources plus simples [53, 56, 71]). Ce problème reste ardu puisque lié à une caractérisation encore plus précise du spectre des opérateurs introduits dans cette thèse.

Bibliographie

[1] AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[2] ALLEN, B., AND MUNRO, I. Self-organizing binary search trees. *Journal of the ACM 25*, 4 (Oct. 1978), 526–535.

[3] APOSTOLICO, A. The myriad virtues of subword trees. In *Combinatorial Algorithms on Words* (1985), A. Apostolico and Z. Galil, Eds., vol. 12 of *NATO Advance Science Institute Series.* Series F : Computer and Systems Sciences, Springer-Verlag, Berlin, pp. 85–96.

[4] APOSTOLICO, A., AND SZPANKOWSKI, W. Self-alignments in words and their applications. *Journal of Algorithms*, 13 (1992), 446–467.

[5] BEDFORD, T., KEANE, M., AND SERIES, C. *Ergodic Theory, Symbolic Dynamics and Hyperbolic Spaces*. Oxford University Press, 1991.

[6] BELL, T., WITTEN, I. H., AND CLEARY, J. G. Modelling for text compression. *ACM Computing Surveys 21*, 4 (1989), 557–591.

[7] BENTLEY, J. A spelling checker. *Communications of the ACM 28*, 5 (May 1985), 456–462.

[8] BENTLEY, J., AND SEDGEWICK, R. Fast algorithms for sorting and searching strings. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (January 1997), SIAM Press, Philadelphia, PA, pp. 360–369. New Orleans.

[9] BENTLEY, J. L. Software exploratorium : the trouble with qsort. *UNIX Review 10*, 2 (1992), 85–93.

[10] BENTLEY, J. L., AND MCILROY, M. D. Engineering a sort function. *Software–Practice and Experience 23*, 11 (1993), 1249–1265.

[11] BOGOMOLNY, E. B., AND CARIOLI, M. Quantum maps from transfer operators. *Physica D 67* (1993), 88–112.

[12] BOURDON, J., NEBEL, M., AND VALLÉE, B. On the stack size of general tries, 2000. (submitted).

[13] BOURDON, J., AND VALLÉE, B. Patricia tries in a dynamical source context, 2000. (in preparation).

[14] BURGE, W. H. An analysis of binary search trees formed from sequences of nondistinct keys. *Journal of the ACM 23*, 3 (July 1976), 451–454.

[15] CLAMPETT, H. A. Randomized binary searching with tree structures. *Communications of the ACM 7*, 3 (Mar. 1964), 163–165.

[16] CLÉMENT, J., FLAJOLET, P., AND VALLÉE, B. The analysis of hybrid trie structures. In *Proceedings of the Ninth Annual ACM–SIAM Symposium on Discrete Algorithms* (Philadelphia, 1998), SIAM Press, pp. 531–539.

[17] CLÉMENT, J., AND ZIMMERMANN, P. An efficient spell-checker based on ternary search tries (provisional title). Research report, Institut National de Recherche en Informatique et en Automatique, 2000. In preparation.

[18] CLÉMENT, J., FLAJOLET, P., AND VALLÉE, B. Dynamical sources in information theory : a general analysis of tries structures. *Algorithmica* (2000), 63 pages. (special issue).

[19] DAUDÉ, H., FLAJOLET, P., AND VALLÉE, B. An average-case analysis of the Gaussian algorithm for lattice reduction. *Combinatorics, Probability and Computing 6* (1997), 397–433.

[20] DE BRUIJN, N. G. *Asymptotic Methods in Analysis*. Dover, 1981. A reprint of the third North Holland edition, 1970 (first edition, 1958).

[21] DEVROYE, L. A probabilistic analysis of the height of tries and of the complexity of triesort. *Acta Informatica 21* (1984), 229–237.

[22] DEVROYE, L. A study of trie-like structures under the density model. *The Annals of Applied Probability 2*, 2 (1992), 402–434.

[23] DIJKSTRA, E. W. *A Discipline of Programming*. Prentice-Hall, Englewood Cliffs, N. J., 1976, ch. 14.

[24] ELLISON, W., AND FRANCE, M. M. *Les nombres premiers*. Publications de l'institut de mathématique de l'université de NANCANGO, IX. Éditions Hermann, 1975.

[25] FAGIN, R., NIEVERGELT, J., PIPPENGER, N., AND STRONG, R. Extendible hashing : A fast access method for dynamic files. *ACM Transactions on Database Systems 4* (1979), 315–344.

[26] FAIVRE, C. Distribution of Lévy's constants for quadratic numbers. *Acta Arithmetica 61*, 1 (1992), 13–34.

[27] FAYOLLE, G., FLAJOLET, P., AND HOFRI, M. On a functional equation arising in the analysis of a protocol for a multi-accessbroadcast channel. *Advances in Applied Probabilities*, 18 (1986), 441–472.

[28] FAYOLLE, G., FLAJOLET, P., HOFRI, M., AND JACQUET, P. Analysis of a stack algorithm for random access communication. *IEEE Transactions on Information Theory IT-31*, 2 (Mar. 1985), 244–254. (Special Issue on Random Access Communication, J. Massey editor).

[29] FELLER, W. *An Introduction to Probability Theory and Its Applications*, vol. 2. John Wiley, 1971.

[30] FILL, J. A., MAHMOUD, H., AND SZPANKOWSKI, W. On the distribution for the duration of a randomized leader election algorithm. *Ann. Appl. Probab.* (1996).

[31] FLAJOLET, GOURDON, AND MARTINEZ. Patterns in random binary search trees. *RSA : Random Structures & Algorithms 11* (1997).

[32] FLAJOLET, P. On approximate counting. In *International Seminar on Modelling and Performance Evaluation Methodology* (Jan. 1983), F. Baccelli and G. Fayolle, Eds., INRIA, Rocquencourt, France, pp. 205–236.

[33] FLAJOLET, P. On the performance evaluation of extendible hashing and trie searching. *Acta Informatica 20* (1983), 345–369.

[34] FLAJOLET, P. Evaluation de protocoles de communication : aspects mathématiques. Technical Report 797, Institut National de Recherche en Informatique et en Automatique, 1988. 22 pages. Main lecture delivered at the *Journée annuelle de la Société Mathématique de France*, Paris, January 1988. Also published by S.M.F., pp. 1–22, 1988.

[35] FLAJOLET, P. Adaptive sampling. In *Encyclopedia of Mathematics*, Kluwer Academic Publishers, Dordrecht, 1997. In press.

[36] FLAJOLET, P., GARDY, D., AND THIMONIER, L. Birthday paradox, coupon collectors, caching algorithms, and self–organizing search. *Discrete Applied Mathematics 39* (1992), 207–229.

[37] FLAJOLET, P., GOURDON, X., AND DUMAS, P. Mellin transforms and asymptotics : Harmonic sums. *Theoretical Computer Science 144*, 1–2 (June 1995), 3–58.

[38] FLAJOLET, P., AND JACQUET, P. Analytic models for tree communication protocols. In *Flow Control of Congested Networks* (1987), A. R. Odoni, L. Bianco, and G. Szegö, Eds., vol. 38 of *NATO Advance Science Institute Series.* Series F : Computer and Systems Sciences, Springer-Verlag, Berlin, pp. 223–234. (Invited Lecture).

[39] FLAJOLET, P., AND MARTIN, G. Probabilistic counting for data base applications. *J. Comput. Syst. Sci*, 31 (1985), 182–209.

[40] FLAJOLET, P., AND PUECH, C. Partial match retrieval of multidimensional data. *Journal of the ACM 33*, 2 (1986), 371–407.

[41] FLAJOLET, P., RÉGNIER, M., AND SOTTEAU, D. Algebraic methods for trie statistics. *Annals of Discrete Mathematics 25* (1985), 145–188. In *Analysis and Design of Algorithms for Combinatorial Problems*, G. Ausiello and M. Lucertini Editors. (Invited Lecture).

[42] FLAJOLET, P., AND SAHEB, N. The complexity of generating an exponentially distributed variate. *Journal of Algorithms 7* (1986), 463–488.

[43] FLAJOLET, P., AND SEDGEWICK, R. Analytic combinatorics. Book in preparation, 1998. (Individual chapters are available as INRIA Research Reports 1888, 2026, 2376, 2956, 3162.).

[44] FLAJOLET, P., AND STEYAERT, J.-M. A branching process arising in dynamic hashing, trie searching and polynomial factorization. In *Automata, Languages and Programming* (1982), M. Nielsen and E. M. Schmidt, Eds., vol. 140 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 239–251. Proceedings of 9th ICALP Colloquium, Aarhus, Denmark, July 1982.

[45] FLAJOLET, P., AND VALLÉE, B. Continued fraction algorithms, functional operators and structure constants. *Theoretical Computer Science*, 194 (1998), 1–34.

[46] FLAJOLET, P., AND VALLÉE, B. Continued fraction algorithms, functional operators and structure constants. In *Proceedings of the Canadian Mathematical Society* (2000), vol. Constructive, Experimental, and Nonlinear Analysis (in the honour of Jonathan Borwein).

[47] FREDKIN, E. Trie memory. *Comm. ACM 3* (1960), 490–499. or 1962 ? ? ?

[48] GONNET, G. H., AND BAEZA-YATES, R. *Handbook of Algorithms and Data Structures : in Pascal and C*, second ed. Addison–Wesley, 1991.

[49] GRABNER, P. J. Searching for losers. *Random Structures and Algorithms 4*, 1 (1993), 99–110.

[50] GROTHENDIECK, A. *Produits tensoriels topologiques et espaces nucléaires*. No. 16 in Memoirs of the American Mathematical Society. A.M.S., Providence, 1955.

[51] GROTHENDIECK, A. La théorie de Fredholm. *Bulletin de la Société Mathématique de France 84* (1956), 319–384.

[52] HOARE, C. A. R. Quicksort. *Computer J. 5*, 1 (1962), 10–15.

[53] JACQUET, P., AND RÉGNIER, M. Trie partitioning process : Limiting distributions. In *CAAP'86* (1986), P. Franchi-Zanetacchi, Ed., vol. 214 of *Lecture Notes in Computer Science*, pp. 196–210. Proceedings of the 11th Colloquium on Trees in Algebra and Programming, Nice France, March 1986.

[54] JACQUET, P., AND SZPANKOWSKI, W. Analysis of digital tries with Markovian dependency. *IEEE Transactions on Information Theory 37*, 5 (1991), 1470–1475.

[55] JACQUET, P., AND SZPANKOWSKI, W. Autocorrelation on words and its applications : analysis of suffix trees by string-ruler approach. *Journal of Combinatorial Theory. Series A 66*, 2 (1994), 237–269.

[56] JACQUET, P., AND SZPANKOWSKI, W. Asymptotic behavior of the Lempel-Ziv parsing scheme and digital search trees. *Theoretical Computer Science 144*, 1–2 (1995), 161–197.

[57] JACQUET, P., AND SZPANKOWSKI, W. Analytical de-Poissonization and its applications. *Theoretical Computer Science 201*, 1-2 (1998), 1–62.

[58] JANSON, S. Private communication.

[59] JANSON, S., AND SZPANKOWSKI, W. Analysis of an asymmetric leader election algorithm. *Electronic Journal of Combinatorics 9* (1997).

[60] KARLIN, S., AND OST, F. Counts of long aligned word matches among random letter sequences. *Adv. Ann. Prob. 19* (1987), 293–351.

[61] KATO, T. *Perturbation Theory for Linear Operators*. Springer-Verlag, New York, 1980.

[62] KIRSCHENHOFER, P., PRODINGER, H., AND SZPANKOWSKI, W. Analysis of a splitting process arising in probabilistic counting and other related algorithms. *Random Structures & Algorithms 9* (1996), 379–402.

[63] KNUTH, D. E. *The Art of Computer Programming*, third ed., vol. 3 : Sorting and Searching. Addison-Wesley, 1998.

[64] KNUTH, D. E., AND YAO, A. C. *Algorithms and complexity*. Academic Press. New York, 1976, ch. The complexity of non-uniform random number generation.

[65] KRASNOSELSKII, M. *Positive solutions of operator equations*. P. Noordhoff, Groningen, 1964.

[66] LARSON, P. A. Dynamic hashing. *BIT 18* (1978), 184–201.

[67] LAZARD, D. On polynomial factorization. In *Proceedings of the European Computer Algebra Conference (EUROCAM '82)* (Marseille, France, Apr. 1982), J. Calmet, Ed., vol. 144 of *LNCS*, Springer-Verlag, pp. 126–134.

[68] LIND, D., AND MARCUS, B. *An introduction to symbolic dynamics and coding*. Cambridge University Press, Cambridge, 1995.

[69] LITWIN, W. Virtual hashing : A dynamically changing hashing. In *Proc. Very Large Data Bases Conf.* (1978), Berlin, Ed., pp. 517–523.

[70] LOTHAIRE, M. *Combinatorics on Words*, vol. 17 of *Encyclopedia of Mathematics and its Applications*. Addison–Wesley, 1983.

[71] MAHMOUD, H. *Evolution of Random Search Trees*. John Wiley, New York, 1992.

[72] MAYER, D. H. Private communication.

[73] MAYER, D. H. Spectral properties of certain composition operators arising in statistical mechanics. *Communications in Mathematical Physics 68* (1979), 1–8.

[74] MAYER, D. H. On composition operators on Banach spaces of holomorphic functions. *Journal of Functional Analysis*, 35 (1980), 191–206.

[75] MCCREIGHT, E. M. A space-economical suffix tree construction. *Journal of the ACM 23*, 2 (1976), 262–272.

[76] MELHORN, K., AND TSAKALADIS, A. Data structures. In *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed., vol. A : Algorithms and Complexity. North Holland, 1990, ch. 6, pp. 301–341.

[77] MORRIS, R. Counting large numbers of events in small registers. *Communications of the ACM vol. 21*, 10 (October 1978), 840–842.

[78] MORRISON, D. R. Patricia-practical algorithm to retrieve information coded in alphanumeric. *JACM 15*, 4 (1968), 514–534.

[79] NEUMANN, J. V. Various techniques used in connection with random digits. In *Von Neumann's Collected Works*, vol. 5. Pergamon. Elmsford. N.Y., 1963, pp. 768–770.

[80] PITTEL, B. Paths in a random digital tree : limiting distributions. *Advances in Applied Probability 18*, 1 (1986), 139–155.

[81] POLLICOTT, M. A complex Ruelle–Perron–Frobenius theorem and two counterexamples. *Ergodic Theory and Dynamical Systems*, 4 (1984), 135–146.

[82] PRODINGER, H. How to select a loser. *Discrete Mathematics 120* (1993), 149–159.

[83] PRODINGER, H. Combinatorics of geometrically distributed random variables : Left-to-right maxima. *Discrete Mathematics 153* (1996), 253–270.

[84] PRODINGER, H. Combinatorics of geometrically distributed random variables : Lengths of ascending runs. *submitted* (2000).

[85] RODEH, M., PRATT, V. R., AND EVEN, S. A linear algorithm for data compression via string matching. *JACM 28*, 1 (jan 1981), 16–24.

[86] RUELLE, D. *Thermodynamic formalism*. Addison Wesley, 1978.

[87] RUELLE, D. *Dynamical Zeta Functions for Piecewise Monotone Maps of the Interval*, vol. 4 of *CRM Monograph Series*. American Mathematical Society, Providence, 1994.

[88] SCHWARTZ, H. *Composition operators in $\mathcal{H}^p$*. PhD thesis, University of Toledo, 1969.

[89] SEDGEWICK, R. Quicksort with equal keys. *SIAM Journal on Computing 6*, 2 (June 1977), 240–267.

[90] SEDGEWICK, R. *Algorithms in C : Fundamentals, Data Structures, Sorting, Searching*, third ed. Addison–Wesley, Reading, Mass., 1988.

[91] SHAPIRO, J. Compact composition operators on spaces of boundary regular holomorphic functions. *Proceedings of the AMS 100* (1997), 49–57.

[92] SHAPIRO, J., AND TAYLOR, P. Compact, nuclear, and Hilbert–Schmidt composition operators on $\mathcal{H}^2$. *Indiana Univiversity Mathematical Journal*, 23 (1973), 471–496.

[93] SLEATOR, D. D., AND TARJAN, R. E. Self-adjusting binary search trees. *JACM 32*, 3 (1985), 652–686.

[94] SZPANKOWSKI, W. The evaluation of an alternative [*sic* !] sum with applications to the analysis of algorithms. *Information Processing Letters 28* (1988), 13–19.

[95] SZPANKOWSKI, W. Some results on V-ary asymmetric tries. *Journal of Algorithms 9* (1988), 224–244.

[96] SZPANKOWSKI, W. On the height of digital trees and related problems. *Algorithmica 6*, 2 (1991), 256–277.

[97] TITCHMARSH, E. C., AND HEATH-BROWN, D. R. *The Theory of the Riemann Zeta-function*, second ed. Oxford Science Publications, 1986.

[98] UKKONEN, E. On-line construction of suffix trees. *Algorithmica 14*, 3 (1995), 249–260.

[99] VALLÉE, B. Algorithms for computing signs of 2×2 determinants : dynamics and average–case analysis. In *Algorithms—ESA '97* (1997), G. W. R. Burkhard, Ed., no. 1136 in Lecture Notes in Computer Science, pp. 486–499. Proceedings of the Fifth European Symposium on Algorithms, Graz, September 1997.

[100] VALLÉE, B. Opérateurs de Ruelle-Mayer généralisés et analyse des algorithmes d'Euclide et de Gauss. *Acta Arithmetica 2*, 81 (1997), 101–144.

[101] VALLÉE, B. Dynamics of the binary euclidean algorithm : Functional analysis and operators. *Algorithmica 22*, 4 (1998), 660–685.

[102] VALLÉE, B. Dynamical sources in information theory : fundamental intervals and word prefixes. *Algorithmica* (2000). (special issue).

[103] WEINER, P. Linear pattern matching algorithms. In *Proceedings of the 14th Annual IEEE Symposium on Switching and Automata Theory* (1973), pp. 1–11.

[104] WELSH, D. *Codes and Cryptography*. Oxford University press, 1990.

[105] ZIMMERMANN, P. Epelle : un logiciel de détection de fautes d'orthographe. Tech. Rep. 2030, INRIA, Septembre 1993.

Annexe A

Source en fraction continue : étude de la série de Dirichlet

Nous détaillons ici quelques calculs pour la source en fraction continue. Nous commençons par rappeler quelques propriétés de l'opérateur de Ruelle-Mayer pour les fractions continues. L'expression de l'entropie de la source est ensuite succinctement calculée. Enfin le calcul de la constante C du corollaire 9.3 est examiné en détail. Enfin, une expression de la série de Dirichlet d'intervalles fondamentaux dans le cas le plus simple $\Lambda(s) \equiv \Lambda^{\langle A\rangle}(\mathrm{Id}, s)$ qui permet de conclure par rapport à la croissance faible de $\Lambda(s)$ sur une bande verticale du plan complexe.

A.1 Propriétés de la source en fraction continue

A.1.1 Propriétés spectrales dominantes

Au voisinage de $s = 1$, on écrit

$$\mathcal{G}_s^k[f](z) = \lambda(s)^k e_s[f]\psi_s(z) + \mathcal{N}_s^k[f](z).$$

Les objets spectraux dominants en $s = 1$ sont connus

$$e_s[f] = \int_0^1 f(t)dt, \qquad \psi_1(z) = \frac{1}{\log 2}\frac{1}{1+z}.$$

La fonction propre dominante de $\mathcal{G}_1$ est nommée mesure de Gauß.

A.1.2 Continuants

Le calcul utilise des propriétés sur la forme très spécifique des branches inverses. À un réel $x \in \mathcal{I} =]0, 1[$, on associe la suite $x_0 = x, x_1, x_2, \ldots, x_k, \ldots$ des itérés de x. Si le k^{e} itéré existe, l'exécution de l'algorithme des fractions continues sur l'entrée x_0 se traduit par un développement en fraction continue

$$x_0 = \cfrac{1}{m_1 + \cfrac{1}{m_2 + \cfrac{1}{\ddots \cfrac{}{m_k + x_k}}}},$$

où les entiers m_j sont supérieurs ou égaux à 1. La relation $x_k = h(x_0)$ définit une homographie h de hauteur k, associée à un k-uplet $(m_1, m_2, \ldots, m_k)$ d'entiers $m_i \geq 1$,

$$h(x) = \cfrac{1}{m_1 + \cfrac{1}{m_2 + \cfrac{1}{\ddots \cfrac{}{m_k + x}}}}.$$

Une telle homographie s'exprime alors à l'aide des *continuants*

$$h(z) = \frac{P_k + zP_{k-1}}{Q_k + zQ_{k-1}},$$

où

$$\begin{array}{rclrcl} Q_k & = & Q_k(m_1, \ldots, m_k), & Q_{k-1} & = & Q_{k-1}(m_1, \ldots, m_{k-1}), \\ P_k & = & Q_{k-1}(m_2, \ldots, m_k), & P_{k-1} & = & Q_{k-2}(m_2, \ldots, m_{k-1}). \end{array}$$

Les polynômes continuants sont définis par récurrence

$$Q_k(m_1, m_2, \ldots, m_k) = m_k Q_{k-1}(m_1, \ldots, m_{k-1}) + Q_{k-2}(m_1, \ldots, m_{k-2}),$$

avec $Q_0 = 1$, $Q_1(m_1) = m_1$. Le polynôme continuant $Q_k(m_1, m_2, \ldots, m_k)$ est aussi la somme de tous les monômes obtenus en barrant deux variables consécutives $m_i m_{i+1}$ dans le produit $m_1 m_2 \cdots m_k$. Les continuants vérifient une propriété de *symétrie*

$$Q_k(m_1, \ldots, m_k) = Q_k(m_k, \ldots, m_1),$$

et l'*identité du déterminant*

$$Q_k P_{k-1} - Q_{k-1} P_k = (-1)^k.$$

L'intervalle fondamental $\mathcal{I}_{\boldsymbol{w}}$ avec $\boldsymbol{w} = m_1 m_2 \cdots m_k$ est de longueur égale à

$$|\mathcal{I}_{\boldsymbol{w}}| = \frac{1}{Q_k(Q_k + Q_{k-1})}.$$

A.2 Croissance modérée de $\Lambda(s)$

La série $\Lambda(s) \equiv \Lambda^{\langle A \rangle}(\mathrm{Id}, s)$ s'exprime grâce aux itérés de l'opérateur de Ruelle Mayer. En effet, on calcule

$$\begin{aligned} \Lambda(s) &= \sum_{\boldsymbol{w} \in \mathcal{M}^*} u_{\boldsymbol{w}}^s \\ &= \sum_{k \geq 0} \sum_{\boldsymbol{w} \in \mathcal{M}^k} \frac{1}{{Q_k}^s (Q_{k-1} + Q_k)^s} \\ &= \sum_{k \geq 0} \sum_{\boldsymbol{w} \in \mathcal{M}^k} \frac{1}{{Q_k}^{2s} (1 + \frac{Q_{k-1}}{Q_k})^s} \\ &= \sum_{k \geq 0} \mathcal{G}_s^k [\frac{1}{(1+x)^s}](0). \end{aligned}$$

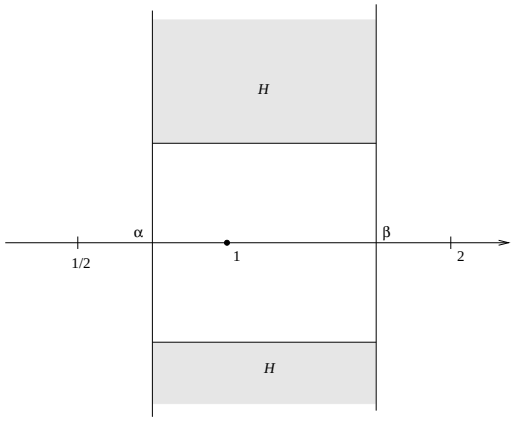


FIG. A.1 – situation pour la proposition A.1.

Formellement, on a

$$\Lambda(s) = (I - \mathcal{G}_s)^{-1}[\frac{1}{(1+x)^s}](0).$$

D'après les propriétés spectrales de $\mathcal{G}_s$, $\Lambda(s)$ possède un pôle simple en $s = 1$. En effet, au voisinage de $s = 1$, $\mathcal{G}_s$ s'écrit

$$\mathcal{G}_s^k[f](z) = \lambda(s)^k \; e_s[f] \; \psi_s(z) \; + \mathcal{N}_s^k[f](z),$$

où

$$\psi_1(z) = \frac{1}{\log 2}\frac{1}{1+z}, \qquad e_1(f) = \int_0^1 f(x)dx.$$

Dans notre cas, $f(z) = \frac{1}{(1+z)^s}$, on en déduit que, dans un voisinage de $s = 1$,

$$\Lambda(s) = \frac{1}{1-\lambda(s)} \; e_s^\star[f](0) \; \psi_s(z) + (I - \mathcal{N}_s)^{-1}[f](0).$$

Le rayon spectral de $\mathcal{N}_s$ étant strictement inférieur à 1, $(I - \mathcal{N}_s)^{-1}$ est analytique en $s = 1$.
Pour appliquer le théorème des sommes harmoniques, il faut encore trouver un réel $\delta < 1$ tel que Λ soit analytique sur $\mathrm{Re}(s) = \delta$ et aussi prouver que Λ lorsque l'on s'éloigne de l'axe réel à l'intérieur d'une bande verticale ne croisse pas trop vite. C'est l'objet de la proposition suivante

PROPOSITION A.1 (CROISSANCE MODÉRÉE DE $\mathbf{\Lambda(s)}$). *$\Lambda(s)$ n'a pas de pôles sur $\left\langle \frac{1}{2}, 1 \right\rangle$ et, dans le domaine $\mathcal{H}$, est en $O(|s|^r)$ pour un $r \geq 0$, avec*

$$\mathcal{H} = \{s \in \mathbb{C} \mid \frac{1}{2} < \alpha < \mathrm{Re}(s) < \beta < 2 \;\; et \;\; Im(s) \geq 4\}.$$

Pour clarifier les choses on peut se référer à la figure A.1.

Preuve. La fonction Λ s'écrit en fonction des continuants

$$\Lambda(s) = \sum_h {u_h}^s = 1 + \sum_{k=1}^{\infty} \sum_{|h|=k} \frac{1}{{Q_k}^s (Q_k + Q_{k-1})^s}.$$

Or d'après une propriété des continuants

$$Q_{k-1}(m_1,\dots,m_{k-1}) = P_k(m_{k-1},\dots,m_1),$$

donc

$$\Lambda(s) = 1 + \sum_{k=1}^{\infty} \sum_{\substack{m_1,\dots,m_k \\ m_i \geq 1}} \frac{1}{{Q_k}^s (Q_k + P_k)^s}.$$

À chaque fraction irréductible c/d de l'intervalle $]0,1[$, on peut associer 2 développements en fraction continue, l'un propre où le dernier quotient vérifie $m_k \geq 1$, l'autre impropre où le dernier quotient vérifie $m_{k+1} = 1$. Ainsi, tout couple (c,d) d'entiers vérifiant $\mathrm{pgcd}(c,d) = 1$, $d \geq 2$ et $0 < c < d$ peut s'écrire de 2 manières différentes comme un couple (P_k, Q_k). Par ailleurs, le nombre 1 s'écrit de manière unique $(1,1)$ et on obtient donc

$$\begin{aligned}\Lambda(s) &= 1 + \frac{1}{2^s} + 2 \sum_{\substack{d\geq 2 \\ 0<c<d \\ \mathrm{pgcd}(c,d)=1}} \frac{1}{d^s(c+d)^s} \\ &= 1 + \frac{1}{2^s} + 2 \sum_{\substack{d\geq 2 \\ 0<c<d \\ \mathrm{pgcd}(c,d)=1}} \frac{1}{d^s(c+d)^s}\end{aligned}$$

On considère les 2 ensembles

$$\begin{aligned}\Omega &= \{(d,c) \in \mathbb{Z}^2 \mid d \geq 2, 0 < c < d\} \\ \Omega' &= \{(d,c) \in \mathbb{Z}^2 \mid d \geq 2, 0 < c < d, \mathrm{pgcd}(c,d) = 1\}.\end{aligned}$$

Tout élément (d,c) de Ω s'écrit de manière unique $(d = ud', c = uc')$ avec $(c',d') \in \Omega'$ et $u > 1$ (c'est la propriété du pgcd).

$$\begin{aligned}\sum_{(d,c)\in\Omega} \frac{1}{d^s(c+d)^s} &= \sum_{u\geq 1} \sum_{(c,d)\in\Omega'} \frac{1}{(ud)^s(uc+ud)^s} \\ &= \sum_{u\geq 1} \frac{1}{u^{2s}} \sum_{(c,d)\in\Omega'} \frac{1}{d^s(c+d)^s} \\ &= \zeta(2s) \sum_{(c,d)\in\Omega'} \frac{1}{d^s(c+d)^s} \\ &= \zeta(2s) \sum_{\substack{d\geq 2 \\ 0<c<d \\ \mathrm{pgcd}(c,d)=1}} \frac{1}{d^s(c+d)^s}\end{aligned}$$

La condition $\mathrm{pgcd}(d,c) = 1$ peut donc être supprimée en divisant par $\zeta(2s)$. On a alors

$$\begin{aligned}\Lambda(s) &= 1 + \frac{1}{2^s} + \frac{2}{\zeta(2s)} \sum_{d\geq 2} \frac{1}{d^s} \sum_{0<c<d} \frac{1}{(c+d)^s} \\ &= 1 + \frac{1}{2^s} + \frac{2}{\zeta(2s)} \sum_{d\geq 2} \frac{1}{d^s} \sum_{d<c<2d} \frac{1}{c^s}\end{aligned}$$

Dans la double somme, on a besoin de bornes non strictes sur c (pour utiliser une majoration par des intégrales). On a

$$\begin{aligned}\sum_{d\geq 2}\frac{1}{d^s}\sum_{d<c<2d}\frac{1}{c^s} &= \sum_{d\geq 1}\frac{1}{d^s}\sum_{d<c<2d}\frac{1}{c^s}\\ &= \sum_{d\geq 1}\frac{1}{d^s}\sum_{d\leq c\leq 2d}\frac{1}{c^s} - \sum_{d\geq 1}\frac{1}{d^s}(\frac{1}{d^s}+\frac{1}{(2d)^s})\\ &= \sum_{d\geq 1}\frac{1}{d^s}\sum_{d\leq c\leq 2d}\frac{1}{c^s} - (1+\frac{1}{2^s})\sum_{d\geq 2}\frac{1}{d^{2s}}\\ &= -(1+\frac{1}{2^s})\zeta(2s)\sum_{d\geq 1}\frac{1}{d^s} + \sum_{d\leq c\leq 2d}\frac{1}{c^s}\end{aligned}$$

En résumé, on a obtenu

$$\begin{aligned}\Lambda(s) &= 1+\frac{1}{2^s}+\frac{2}{\zeta(2s)}(\sum_{d\geq 1}\frac{1}{d^s}\sum_{d\leq c\leq 2d}\frac{1}{c^s} - (1+\frac{1}{2^s})\zeta(2s))\\ &= -1-\frac{1}{2^s}+\frac{2}{\zeta(2s)}\sum_{d\geq 1}\frac{1}{d^s}\sum_{d\leq c\leq 2d}\frac{1}{c^s}\end{aligned}$$

La formule de sommation d'Euler-Maclaurin permet d'évaluer

$$\sum_{d\leq c\leq 2d}\frac{1}{c^s}.$$

THÉORÈME A.2. *Soit f une fonction à valeurs dans $\mathbb{C}$ définie sur l'intervalle $[a,b]$, où a et b sont des entiers. Si f est dérivable et à dérivée continue sur l'intervalle $[a,b]$, alors on a*

$$\sum_{a\leq k\leq b} f(k) = \int_a^b f(x)\,dx \;+\; \frac{f(a)+f(b)}{2} \;+\; \int_a^b \left(\{x\}-\frac{1}{2}\right) f'(x)\,dx,$$

où $\{x\}$ est la partie fractionnaire de x.

Ainsi, on a

$$\begin{aligned}\sum_{d\leq c\leq 2d}\frac{1}{c^s} &= \int_d^{2d}\frac{dx}{x^s}+\frac{1}{2}(\frac{1}{d^s}+\frac{1}{(2d)^s})+\int_d^{2d}(\{x\}-\frac{1}{2})\frac{-s}{x^{s+1}}\,dx\\ &= \frac{1}{s-1}(1-2^{1-s})\frac{1}{d^{s-1}}+\frac{1}{2}(1+2^{-s})\frac{1}{d^s}-s\int_d^{2d}(\{x\}-\frac{1}{2})\frac{1}{x^{s+1}}\,dx\end{aligned}$$

On obtient finalement

$$\begin{aligned}\Lambda^{\langle A\rangle}(\mathrm{Id},s) &= -1-2^{-s}+\frac{2}{\zeta(2s)}\left(\frac{1-2^{1-s}}{s-1}\zeta(2s-1)+\frac{1+2^{-s}}{2}\zeta(2s)\right)-R(s)\\ &= 2\,\frac{1-2^{1-s}}{s-1}\frac{\zeta(2s-1)}{\zeta(2s)}-R(s) \qquad \text{(A.1)}\end{aligned}$$

avec

$$R(s) = \frac{2s}{\zeta(2s)}\sum_{d\geq 1}\frac{1}{d^s}\int_d^{2d}(\{x\}-\frac{1}{2})\frac{1}{x^{s+1}}\,dx.$$

Posant $\sigma = \mathrm{Re}(s)$, on remarque

$$\left| \int_d^{2d} (\{x\} - \frac{1}{2}) \frac{1}{x^{s+1}}\, dx \right| \leq \int_d^{2d} \left| \frac{1}{x^{s+1}} \right| dx = \int_d^{2d} \frac{1}{x^{\sigma+1}}\, dx = \frac{1}{\sigma}(2^{-\sigma} - 1)\frac{1}{d^\sigma}$$

d'où

$$|R(s)| \leq \left| \frac{2s}{\zeta(2s)} \right| \sum_{d \geq 1} \left| \frac{1}{d^s} \right| \frac{1}{\sigma}(2^{-\sigma} - 1)\frac{1}{d^\sigma} = 2\,\frac{|s|}{\sigma}(2^{-\sigma} - 1)\frac{\zeta(|2\sigma|)}{|\zeta(2s)|}$$

Ainsi $R(s)$ est analytique pour $\mathrm{Re}(s) > \frac{1}{2}$. On voit donc que $\Lambda(s)$ n'a pas de pôles pour $\frac{1}{2} < \mathrm{Re}(s) < 1$.
La deuxième partie de la preuve peut être faite grâce à des majorations sur un domaine approprié de $\zeta(2s-1)^{-1}$ et de $\zeta(2s)$ (d'après [24]). □

A.3 Calcul de l'entropie $h(\mathsf{CF})$

On calcule $h(\mathsf{CF}) = -\lambda'(1)$ grâce aux propriétés spectrales dominantes de $\mathcal{G}_s$ en considérant les résidus de

$$\frac{-1}{\lambda'(1)} \frac{1}{\log 2} \int_0^1 dt = \mathrm{Res}\left[(I - \mathcal{G}_s)^{-1}[1](0) \right]_{s=1} = \mathrm{Res}\left[\sum_{\boldsymbol{w} \in \mathcal{M}^*} \frac{1}{Q_k^{2s}} \right]_{s=1}.$$

Or par un calcul du même type (mais beaucoup plus simple) que celui conduisant à l'expression (A.1), on a

$$\sum_{\boldsymbol{w} \in \mathcal{M}^*} \frac{1}{Q_k^{2s}} = \sum_{\substack{c,d \\ c<d \\ \mathrm{pgcd}(c,d)=1}} \frac{1}{d^{2s}} = \frac{\zeta(2s-1)}{\zeta(2s)} - 1,$$

où $\zeta(s) = \sum_{k \geq 1} 1/k^s$ est la fonction ζ de Riemann. Le résidu de cette expression en $s = 1$ est $1/\zeta(2) = 6/\pi^2$. On obtient finalement $-\lambda'(1) = \dfrac{\pi^2}{6 \log 2}$.

A.4 Cas la longueur de cheminement pour les tries-ABR

On rappelle l'expression pour la longueur de cheminement du trie-ABR du corollaire 9.3.

$$P_B(n) \sim Cn \log n \text{ avec } C := \frac{3}{\pi^2} \left[1 + 4 \sum_{k \geq 2} \frac{1}{k^2 - 1} \log \frac{k+1}{2} \right].$$

Le but de cette section est de montrer un exemple de calcul de constante pour la source en fraction continue. Le calcul de C ne dépend pas de la distribution choisie F (voir la section 9.1.3 et les théorèmes de la section 8.2) ; on peut donc supposer une distribution uniforme. La série de Dirichlet s'écrit

$$\Lambda^{\langle B \rangle}(\mathrm{Id}, s) = 2 \sum_{\boldsymbol{w} \in \mathcal{M}^*} \sum_{i<j} \frac{u_{\boldsymbol{w} \cdot i} u_{\boldsymbol{w} \cdot j}}{U_{\boldsymbol{w} \cdot [i,j]}^{2-s}}.$$

En considérant les continuants correspondant à un mot $\boldsymbol{w} = m_1 m2 \cdots m_k$ de longueur k, on calcule alors

$$u_{\boldsymbol{w}\cdot i} = \frac{1}{Q_k^2(\frac{Q_{k-1}}{Q_k}+i)(\frac{Q_{k-1}}{Q_k}+i+1)}, \quad U_{\boldsymbol{w}\cdot[i,j]} = \frac{(j-i+1)}{Q_k^2(\frac{Q_{k-1}}{Q_k}+i)(\frac{Q_{k-1}}{Q_k}+j+1)}.$$

On calcule

$$\mathcal{G}_s^k[f](0) = \sum_{(m_1,\ldots,m_k)\in\mathbb{N}^k} \frac{1}{Q_k^{2s}}\, f\left(\frac{P_k}{Q_k}\right) = \sum_{(m_1,\ldots,m_k)\in\mathbb{N}^k} \frac{1}{Q_k^{2s}}\, f\left(\frac{Q_{k-1}}{Q_k}\right)$$

où la deuxième égalité provient de la relation de symétrie des continuants. On peut donc écrire par identification

$$\begin{aligned}\Lambda^{\langle B\rangle}(\mathrm{Id}, s) &= 2\sum_{\boldsymbol{w}\in\mathcal{M}^*}\sum_{i<j} \frac{1}{(j-i+1)^{2-s}}\frac{1}{Q_k^{2s}}\frac{(\frac{Q_{k-1}}{Q_k}+i)^{1-s}(\frac{Q_{k-1}}{Q_k}+j+1)^{1-s}}{(\frac{Q_{k-1}}{Q_k}+i+1)(\frac{Q_{k-1}}{Q_k}+j)}\\ &= 2(I-\mathcal{G}_s)^{-1}[f_{ij}(z)](0)\end{aligned}$$

avec

$$f_{ij}(z) = \frac{1}{(j-i+1)^{2-s}}\frac{(z+i)^{1-s}(z+j+1)^{1-s}}{(z+i+1)(z+j)}.$$

Le résidu en $s=1$ est

$$\frac{-1}{\lambda'(1)}\frac{2}{\log 2}R \quad \text{avec} \quad R = \int_0^1 \left(\sum_{i<j}\frac{1}{(j-i+1)}\frac{1}{(t+i+1)(t+j)}\right)dt.$$

On calcule (en posant $k=j+1$)

$$R = \sum_{i=1}^{\infty}\sum_{k=1}^{\infty}\int_0^1 \frac{1}{k+1}\frac{1}{(t+i+1)(t+i+k)}dt.$$

Or la décomposition en éléments simples donne

$$\frac{1}{(t+i+1)(t+i+k)} = \begin{cases}\dfrac{1}{k-1}\left(\dfrac{1}{t+i+1}-\dfrac{1}{t+i+k}\right)\dfrac{1}{2}\displaystyle\int_0^1\frac{1}{(t+i+1)^2}\,dt & \text{si } k\neq 1,\\ \dfrac{1}{(t+i+1)^2} & \text{si } k=i.\end{cases}$$

Finalement, la quantité I s'écrit

$$\begin{aligned}R &= \sum_{i=1}^{\infty}\frac{1}{2}\int_0^1\frac{1}{(t+i+1)^2}\,dt + \sum_{i=0}^{\infty}\sum_{k=2}^{\infty}\frac{1}{(k+1)(k-1)}\left[\int_0^1\frac{1}{t+i+1}dt - \int_0^1\frac{1}{t+i+k}dt\right]\\ &= \frac{1}{2}\int_2^{\infty}\frac{1}{t^2}\,dt + \sum_{k=2}^{\infty}\frac{1}{k^2+1}\int_2^{k+1}\frac{dt}{t}\\ &= \frac{1}{4} + \sum_{k=2}^{\infty}\frac{1}{k^2-1}\log\frac{k+1}{2}.\end{aligned}$$

Annexe B

Chaîne de Markov

On étudie ici les constantes obtenue lors de l'analyse des tries (en particulier entropie, et constantes liées à l'hybridation) dans le cas d'une chaîne de Markov avec une distribution uniforme. De même que pour la source en fraction continue, on peut supposer la distribution uniforme puisque la distribution n'intervient pas dans le calcul des constantes de cette annexe (voir le chapitre 8).
On définit une chaîne de Markov M à l'aide d'une matrice de transition $P = (p_{i|j})$ et d'un vecteur de probabilités initiales $\mathbf{p} = (p_{i|0})$ (où l'état 0 correspond à l'état initial).
Les séries de Dirichlet qui interviennent sont les suivantes :

$$\begin{aligned}
\Lambda_k^{\langle A\rangle}(F,s) &= \sum_{\boldsymbol{w}\in\mathcal{M}^k} u_{\boldsymbol{w}}^s \\
\Lambda^{\langle A\rangle}(F,s) &= \sum_{\boldsymbol{w}\in\mathcal{M}^*} u_{\boldsymbol{w}}^s, \\
\Lambda^{\langle B\rangle}(F,s) &= 2\sum_{\boldsymbol{w}\in\mathcal{M}^*}\sum_{\substack{(i,j)\in\mathcal{M}^2\\ i<j}} u_{\boldsymbol{w}\cdot i}\, u_{\boldsymbol{w}\cdot j}\, U_{\boldsymbol{w}\cdot[i,j]}^{s-2}, \\
\Lambda^{\langle L\rangle}(F,s) &= \sum_{\boldsymbol{w}\in\mathcal{M}^*}\sum_{i\in\mathcal{M}} U_{\boldsymbol{w}\cdot[>i]}\, u_{\boldsymbol{w}\cdot i}^{s-1}.
\end{aligned}$$

On trouve une expression alternative des séries ci-dessus pour une chaîne de markov (par récurrence) :

$$\begin{aligned}
\Lambda_k^{\langle A\rangle}(\mathrm{Id},s) &= {}^t\mathbf{e}P_s^{k-1}\mathbf{p}_s \qquad (k\geq 1)\\
\Lambda^{\langle A\rangle}(\mathrm{Id},s) &= 1+{}^t\mathbf{e}(I-P_s)^{-1}\mathbf{p}_s,\\
\Lambda^{\langle B\rangle}(\mathrm{Id},s) &= 1+{}^t\mathbf{b}_s(I-P_s)^{-1}\mathbf{p}_s,\\
\Lambda^{\langle L\rangle}(\mathrm{Id},s) &= 1+{}^t\mathbf{l}_s(I-P_s)^{-1}\mathbf{p}_s.
\end{aligned}$$

Ces expressions font intervenir le vecteur unité ${}^t\mathbf{e}$, la matrice $P_s = (p_{i|j}^s)$, le vecteur $p_s = (p_{i|0}^s)$ et les vecteurs $\mathbf{b}_s = (b_s[k])$ et $\mathbf{l}_s = (l_s[k])$ de composantes

$$b_s[k] = 2\sum_{i<j} p_{i|k}p_{j|k}\, P_{[i,j]|k}^{s-2}, \qquad l_s[k] = \sum_i P_{[>i]|k}\, p_{i|k}^{s-1},$$

où comme dans toute cette thèse $P_{[i,j]|k} = p_{i|k}+\cdots+p_{j|k}$ et $P_{[>i]|k} = \sum_{\ell>i} p_{\ell|k}$.

Soit $\boldsymbol{\pi}_s$ le vecteur propre associé à la valeur propre dominante $\lambda(s)$. On a

$$P_s \boldsymbol{\pi}_s = \lambda(s) \boldsymbol{\pi}_s.$$

Le vecteur $\boldsymbol{\pi}_s$ est de plus normalisé en $s = 1$ puisqu'il s'agit alors d'un vecteur de probabilités dont la somme des composantes vaut 1.

LEMME B.1 (ENTROPIE D'UNE CHAÎNE DE MARKOV). *L'entropie de la chaîne de Markov est*

$$h(\mathsf{M}) = -\lambda'(1) = -\sum_k \pi_k \sum_i p_{i|k} \log p_{i|k}. \tag{B.1}$$

Preuve. Par définition, l'entropie de la chaîne de Markov M est égale à

$$\begin{aligned} h(\mathsf{M}) &= \lim_{n\to\infty} -\frac{1}{n} \sum_{\boldsymbol{w}\in\mathcal{M}^n} u_{\boldsymbol{w}} \log u_{\boldsymbol{w}} \\ &= \lim_{n\to\infty} -\frac{1}{n} \frac{\partial}{\partial s} \left(\sum_{\boldsymbol{w}\in\mathcal{M}^n} u_{\boldsymbol{w}}^s \right)_{s=1} \\ &= \lim_{n\to\infty} -\frac{1}{n} {}^t\mathbf{e} \frac{\partial}{\partial s} (P_s^{n-1} \mathbf{p}_s)_{s=1}. \end{aligned}$$

Or on calcule la dérivée

$$\frac{\partial}{\partial s}(P_s^{n-1}\mathbf{p}_s) = (n-1)\frac{\partial P_s}{\partial s} P_s^{n-2}\mathbf{p}_s + P_s^{n-1}\frac{\partial \mathbf{p}_s}{\partial s}.$$

En $s = 1$, on obtient

$$h(\mathsf{M}) = -{}^t\mathbf{e} \left(\frac{\partial P_s}{\partial s} \right)_{s=1} \boldsymbol{\pi}_1,$$

où $\boldsymbol{\pi}_1$ est le vecteur stationnaire de la chaîne de Markov. On retrouve ainsi le résultat bien connu

$$h(\mathsf{M}) = -\sum_k \pi_k \sum_i p_{i|k} \log p_{i|k}.$$

Par ailleurs en dérivant par rapport à s la relation $P_s \boldsymbol{\pi}_s = \lambda(s)\boldsymbol{\pi}_s$, on a

$$\frac{\partial P_s}{\partial s}\boldsymbol{\pi}_s + P_s \frac{\partial \boldsymbol{\pi}_s}{\partial s} = \lambda'(s)\boldsymbol{\pi}_s + \lambda(s)\frac{\partial \boldsymbol{\pi}_s}{\partial s}.$$

En multipliant à gauche par ${}^t\mathbf{e}$ (la transposée du vecteur unité), l'équation précédente en $s = 1$ s'écrit

$${}^t\mathbf{e} \left(\frac{\partial P_s}{\partial s} \right)_{s=1} \boldsymbol{\pi}_1 + {}^t\mathbf{e} P_1 \left(\frac{\partial \boldsymbol{\pi}_s}{\partial s} \right)_{s=1} = \lambda'(s) {}^t\mathbf{e}\boldsymbol{\pi}_1 + \lambda(1) \left(\frac{\partial \boldsymbol{\pi}_s}{\partial s} \right)_{s=1}.$$

En tenant compte des simplifications dues au fait que P_1 est une matrice de transition et $\boldsymbol{\pi}_1$ un vecteur de probabilités, il reste

$${}^t\mathbf{e} \left(\frac{\partial P_s}{\partial s} \right)_{s=1} \boldsymbol{\pi}_1 = \lambda'(1),$$

ce qui termine la démonstration. □

Le lemme suivant permet d'expliciter le résidu des séries $\Lambda^{\langle\cdot\rangle}(s, \mathrm{Id})$.

LEMME B.2. *Le résidu de la k^e composante de $(I - P_s)^{-1}\mathbf{p}_s$ est égal à*

$$-\frac{1}{\lambda'(1)}\pi_k,$$

où π_k est la k^e composante du vecteur stationnaire $\boldsymbol{\pi}_1$.

Démonstration. On utilise la relation de décomposabilité (projection sur le sous-espace propre dominant) valable dans un voisinage de l'axe réel $P_s\mathbf{p}_s = \lambda(s)\boldsymbol{\pi}_s + N_s\mathbf{p}_s$. On obtient donc

$$P_s^k\mathbf{p}_s = \lambda(s)^k\boldsymbol{\pi}_s + N_s^k\mathbf{p}_s,$$

et donc

$$(I - P_s)^{-1}\mathbf{p}_s = \frac{1}{1-\lambda(s)}\boldsymbol{\pi}_s + (I - N_s)^{-1}\mathbf{p}_s.$$

Or $(I - N_s)^{-1}$ est holomorphe au voisinage de $s = 1$, on en déduit le résidu de $(I - P_s)^{-1}\mathbf{p}_s$

$$-\frac{1}{\lambda'(1)}\boldsymbol{\pi}_1.$$

□

Enfin la proposition donne les résidus des séries de Dirichlet pour une chaîne de Markov en $s = 1$

PROPOSITION B.3 (RÉSIDUS DES SÉRIES DE DIRICHLET POUR UNE CHAÎNE DE MARKOV). *Les résidus des séries de Dirichlet pour une chaîne de Markov* M *de matrice de transition $P = (p_{i|j})$ et de vecteur stationnaire $\boldsymbol{\pi} = (\pi_k)$ sont*

$$\begin{aligned}
\mathrm{Res}(\Lambda^{\langle A\rangle}(\mathrm{Id}, s), s = 1) &= \frac{1}{h(\mathsf{M})} \\
\mathrm{Res}(\Lambda^{\langle B\rangle}(\mathrm{Id}, s), s = 1) &= \frac{2}{h(\mathsf{M})}\left(\sum_k \pi_k \sum_{i<j} \frac{p_{i|k}p_{j|k}}{P_{[i,j]|k}}\right) \\
\mathrm{Res}(\Lambda^{\langle L\rangle}(\mathrm{Id}, s), s = 1) &= \frac{1}{h(\mathsf{M})}\left(\sum_k \pi_k \sum_i P_{[>i]|k}\right),
\end{aligned}$$

où $h(\mathsf{M}) = -\sum_k \pi_k \sum_i p_{i|k}\log p_{i|k}$ est l'entropie.

Annexe C

Extension aux bucket-tries

On considère souvent pour des applications de type bases de données [25, 66] des variantes de tries appelées b-tries ou «bucket tries» (voir la section 2.2.1), où la règle de construction est un peu relâchée.
Le cadre d'analyse est exactement le même. Nous précisons ici juste les conclusions correspondantes.

COROLLAIRE C.1. *Pour les bucket tries de capacité b, la hauteur obéit une loi de type double exponentielle*

$$\lim_{n\to\infty} \sup_{k\geq 0} \left|\Pr\{h_n \leq k\} - \exp[-\rho_b\, \lambda(b+1)^k\, n^{b+1}]\right| = 0,$$

avec $\rho_b > 0$ et la valeur moyenne de la hauteur satisfait

$$\mathrm{E}[h_n] \sim \frac{b+1}{|\log \lambda(b+1)|} \log n.$$

La taille, défini comme le nombre de nœuds internes de l'arbre a pour espérance

$$S(n) \approx \frac{1}{b\, h(\mathsf{S})}\, n.$$

Les différents types de longueurs de cheminement peuvent être définis et s'analysent de la même façon : En moyenne, une branche de l'arbre est de hauteur $\log n/h(\mathsf{S})$ et diffère de celle d'un trie standard seulement en $O(1)$. On remarque que la distribution de la hauteur change radicalement. La formule pour la taille exprime le fait que les pages sont utilisées avec un taux égal à $h(\mathsf{S})$.

Résumé. Les arbres digitaux, également connus sous le nom de “tries” sont une structure de donnée générique et flexible qui permet d’implanter des dictionnaires construits sur des ensembles de mots. Nous donnons une analyse de trois représentations principales de ces arbres, les arbres-tableaux, les arbres-listes, et les arbres ternaires de recherche. La taille et les coûts de recherche de ces représentations sont analysés précisément en moyenne, tandis qu’une analyse en distribution de la hauteur est obtenue. Le modèle unificateur d’analyse est celui des “sources dynamiques”, lesquelles recouvrent les modèles classiques comme les sources sans mémoire (à symboles indépendants), les chaînes de Markov finies, et les densités initiales non uniformes. Les propriétés probabilistes des principaux paramètres de taille, longueur de cheminement et hauteur apparaissent liées à deux caractéristiques fondamentales de la source : l’entropie et la probabilité de coïncidence. Ces caractéristiques se trouvent elle-mêmes reliées aux propriétés spectrales d’opérateurs de transfert du type introduit par Ruelle.

Mots clés. Théorie de l’information, sources dynamiques, analyse d’algorithmes, arbres digitaux, tries, arbres ternaires de recherche, opérateur de transfert, fractions continues.

Abstract. Digital trees, also known as tries, are a general purpose flexible data structure that implements dictionaries built on sets of words. An analysis is given of three major representations of tries in the form of array-tries, list tries, and bst-tries (“ternary search tries”). The size and the search costs of the corresponding representations are analysed precisely in the average case, while a complete distributional analysis of height of tries is given. The unifying data model used is that of dynamical sources and it encompasses classical models like those of memoryless sources with independent symbols, of finite Markov chains, and of nonuniform densities. The probabilistic behaviour of the main parameters, namely size, path length, or height, appears to be determined by two intrinsic characteristics of the source : the entropy and the probability of letter coincidence. These characteristics are themselves related in a natural way to spectral properties of specific transfer operators of the Ruelle type.

Keywords. Information theory, dynamical sources, analysis of algorithms, digital trees, tries, ternary search tries, transfer operator, continued fractions.

Groupe de Recherche en Informatique, Image, Instrumentation de Caen
CNRS UPRESA 6072