Mémoire d'habilitation à diriger des recherches
présenté le 22 septembre 2017.

Spécialité :
Informatique

Combinatoire énumérative et analytique en Logique Propositionnelle et en Théorie de la Concurrence :

Vers une quantification de l'expressivité des modèles

Antoine Genitrini

Rapporteurs :

M. Michael Drmota	Professeur	Technische Universität Wien (Vienne)
M. Arnaud Durand	Professeur	Université Denis Diderot (Paris)
M. Conrado Martínez	Professeur	Universitat Politècnica de Catalunya (Barcelone)

Examinateurs :

Mme Béatrice Bérard	Professeur	Université Pierre et Marie Curie (Paris)
M. Olivier Bodini	Professeur	Université Paris-Nord (Paris)
M. Alain Denise	Professeur	Université Paris-Sud (Paris)
M. Jean Mairesse	Directeur de Recherche CNRS	Université Pierre et Marie Curie (Paris)
M. Cyril Nicaud	Professeur	Université Paris-Est, Marne-la-Vallée (Paris)
Mme Michèle Soria	Professeur	Université Pierre et Marie Curie (Paris)

Avant-propos

J'ai choisi d'écrire ce document, qui constitue le mémoire pour mon habilitation à diriger des recherches, en m'appuyant sur mes deux domaines de recherche principaux : les études combinatoires et algorithmiques de la logique booléenne d'un côté et de la théorie de la concurrence de l'autre côté. Il n'y a a priori pas de relation directe entre ces deux disciplines. Néanmoins, la nature des études comportent des similarités. En effet, l'idée originelle est basée sur l'étude de la sémantique des modèles, qui aussi bien en logique qu'en théorie des langages, consiste à relier les structures syntaxiques (très souvent des arbres de syntaxe abstraits) et leur interprétation. Nous pensons par exemple aux propositions logiques interprétées comme des fonctions booléennes, ou des arbres de preuves. Par ailleurs, mes approches des problèmes sont unifiées par les techniques utilisées, la combinatoire analytique, et par la nature des questions qui me préoccupent : *comment peut-on interpréter et quelle information peut-on obtenir d'une grande structure construite dans l'un des domaines ?* Cette question, énoncée de manière très informelle, fera apparaître de nombreux choix soit de modèles soit d'études qui seront dévoilés au fil de votre lecture.

Plus précisément, mon but consiste à interpréter les structures syntaxiques et sémantiques en tant que classes combinatoires pour obtenir des informations quantitatives et algorithmiques, grâce aux outils de combinatoire analytique.

Dans ce document, la démarche et la progression sont davantage mis en avant par rapport aux résultats. En particulier, il n'y a aucune preuve exhaustive, seules les idées-clés qui décrivent l'approche sont exposées. Je souhaite ici présenter les intuitions qui permettent d'appréhender les objets combinatoires dans ces contextes, tout en indiquant les pointeurs pour trouver les résultats formalisés avec leur preuves exhaustives.

C'est un mémoire de combinatoire appliquée à deux problèmes de l'informatique : la logique et théorie de la concurrence. De plus, nous allons nous concentrer sur les modèles combinatoires davantage que sur les applications. Pour ces dernières, nos articles sont plus détaillés. De même, les mémoires de thèse [Die17] ou d'habilitation de Matthieu et Frédéric seront plus fournis sur ce point précis.

J'ai souhaité mettre en avant dans ce mémoire, les évolutions des modèles, en expliquant les apports de chaque caractéristique sur les résultats. J'ai mis en avant les nombreux cas, pour lesquels un modèle plus complexe ne modifie pas fondamentalement les résultats obtenus, ou les approches permettant d'obtenir ces résultats. Ces études, itératives, par brique de plus en plus complexe permettent en conséquence aussi de comprendre la combinatoire d'un modèle plus évolué et les raisons essentielles qui font qu'une approche ou un algorithme ne peut finalement pas s'adapter, mais qu'il faut les reconsidérer intégralement pour répondre au nouveau problème.

L'intérêt pour l'étude de la logique d'un point de vue quantitatif a commencé dès ma thèse et certaines questions, parmi les plus délicates, restent toujours en suspens, et continuent d'animer de régulières réunions de travail avec plusieurs collègues. Je souhaite au sein de ce mémoire présenter la démarche que nous avons suivie afin de compléter et enrichir petit-à-petit nos modèles. Par conséquent, il me semble naturel de rappeler les résultats que j'ai prouvés au cours de ma thèse, de 2006 à 2009. Ainsi, dans la Partie I, le Chapitre 2 ainsi que les deux Sections 3.1 et 3.4 du Chapitre 3 contiennent des résultats prouvés dans ma thèse [28]. Les autres résultats de la Partie I concernant la logique ont été prouvés plus tardivement.

Avec Olivier et Frédéric, nous avons commencé à réfléchir à relier syntaxe et sémantique de la concurrence lors de mon intégration à l'UPMC en 2010. Plusieurs caractéristiques rendent ces études particulièrement riches. Tout d'abord, de nombreuses approches pourraient être adaptées directement à la théorie des ordres partiels. Par ailleurs, d'un point de vue des modèles considérés, ils font intervenir des contraintes de croissance d'étiquetage dont la combinatoire et l'algorithmique sont relativement peu développées.

Au final, ce mémoire n'a donc pas pour unique but de synthétiser mes derniers articles de recherche. Il contient plutôt les idées de ma recherche pendant thèse, après ma thèse et à venir. Je souhaite faire ressortir

- l'approche globale que j'ai suivie, de l'exemple-jouet aux modèles de plus en plus complexes
- les points d'accroches d'un modèle plus évolué qui font que les méthodes précédentes ne s'adaptent pas
- les algorithmes génériques qui résolvent les questions dans les modèles les plus évolués (et donc les plus simples également, même si moins efficaces que des méthodes ad hoc).

Par ailleurs, le mémoire n'est pas exhaustif au niveau des citations : ce sont essentiellement les livres et articles, directement reliés et qui enrichissent mes propos qui sont pointés.

Enfin, avant de commencer de façon plus formelle, je souhaite indiquer la liste de mes collègues et amis avec qui je partage de réels moment de plaisir face à mon tableau blanc, et ce, depuis plus de 10 ans maintenant. J'ai choisi la simplicité du classement alphabétique pour les nommer. Olivier Bodini ; Matthieu Dien ; Xavier Fontaine ; Hervé Fournier ; Danièle Gardy ; Bernhard Gittenberger ; Hsien-Kuei Hwang ; Manuel Kauers ; Jakub Kozik ; Veronika Daxner (Kraus) ; Cécile Mailler ; Grzegorz Matecki ; Frédéric Peschanski ; Nicolas Rolin ; Michael Wallner et Marek Zaionc.

Bonne lecture.

Liste de publications

Voilà la liste de mes publications. Les articles sont téléchargeables en format `pdf` à l'url :
`www.lip6.fr/Antoine.Genitrini`.

Revues internationales avec comité de lecture

[1] H. Fournier, D. Gardy, A. Genitrini, and M. Zaionc. Tautologies over implication with negative literals. *Mathematical Logic Quarterly*, 56(4) :388–396, 2010.

[2] H. Fournier, D. Gardy, A. Genitrini, and B. Gittenberger. The fraction of large random trees representing a given boolean function in implicational logic. *Random Structures and Algorithms*, 40(3) :317–349, 2012.

[3] A. Genitrini and J. Kozik. In the full propositional logic, 5/8 of classical tautologies are intuitionistically valid. *Annals of Pure and Applied Logic*, 163(7) :875–887, 2012.

[4] A. Genitrini, B. Gittenberger, V. Kraus, and C. Mailler. Probabilities of Boolean Functions Given by Random Implicational Formulas. *Electronic Journal of Combinatorics*, 19(2) :P2.37, 20 pages, (electronic), 2012.

[5] A. Genitrini, B. Gittenberger, V. Kraus, and C. Mailler. Associative and commutative tree representations for boolean functions. *Theoretical Computer Science*, 570 :70–101, 2015.

[6] O. Bodini, A. Genitrini, and N. Rolin. Pointed versus Singular Boltzmann Samplers : a Comparative Analysis. *PUre Mathematics and Applications*, 25(2) :115–131, 2015.

[7] A. Genitrini and C. Mailler. Generalised and Quotient Models for Random And/Or Trees and Application to Satisfiability. *Algorithmica*, 1 :1–33, 2016.

[8] O. Bodini, A. Genitrini, and F. Peschanski. A Quantitative Study of Pure Parallel Processes. *Electronic Journal of Combinatorics*, 23(1) :P1.11, 39 pages, (electronic), 2016.

[9] V. Daxner, A. Genitrini, B. Gittenberger, and C. Mailler. The Relation between Tree Size Complexity and Probability for Boolean Functions Generated by Uniform Random Trees. *Applicable Analysis and Discrete Mathematics*, 10(2) :408–446, 2016.

[10] O. Bodini, A. Genitrini, and N. Rolin. Extended boxed product and application to synchronized trees. *Electronic Notes in Discrete Mathematics*, 59 :189–202, 2017.

Actes de colloques internationaux avec comité de lecture

[11] H. Fournier, D. Gardy, A. Genitrini, and M. Zaionc. Classical and intuitionistic logic are asymptotically identical. In Springer-Verlag, editor, *Annual Conference on Computer Science Logic (CSL)*, pages 177–193, 2007.

[12] A. Genitrini, J. Kozik, and M. Zaionc. Intuitionistic vs. classical tautologies, quantitative comparison. In Springer-Verlag, editor, *TYPES*, pages 100–109, 2007.

[13] H. Fournier, D. Gardy, A. Genitrini, and B. Gittenberger. Complexity and limiting ratio of boolean functions over implication. In Springer-Verlag, editor, *33rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 347–362, 2008.

[14] A. Genitrini, J. Kozik, and G. Matecki. On the density and the structure of the peirce-like formulae. In *5th Colloquium on Mathematics and Computer Science : Algorithms, Trees, Combinatorics and Probabilities*, pages 461–474, 2008.

[15] A. Genitrini and J. Kozik. Quantitative comparison of intuitionistic and classical logics - full propositional system. In Springer-Verlag, editor, *International Symposium on Logical Foundations of Computer Science (LFCS)*, pages 280–294, 2009.

[16] H. Fournier, D. Gardy, and A. Genitrini. Balanced And/Or trees and linear threshold functions. In *6th SIAM Workshop on Analytic and Combinatorics (ANALCO)*, pages 51–57, 2009.

[17] A. Genitrini and B. Gittenberger. No Shannon effect on probability distributions on Boolean functions induced by random expressions. In *21st International Meeting on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AofA)*, pages 303–316, 2010.

[18] O. Bodini, A. Genitrini, and F. Peschanski. Enumeration and Random Generation of Concurrent Computations. In *23rd International Meeting on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AofA)*, pages 83–96, 2012.

[19] O. Bodini, A. Genitrini, and F. Peschanski. The combinatorics of non-determinism. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 24 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 425–436, 2013.

[20] A. Genitrini, B. Gittenberger, and C. Mailler. No shannon effect induced by and/or trees. In *25th International Meeting on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AofA)*, pages 109–120, 2014.

[21] A. Genitrini and C. Mailler. Equivalence classes of random Boolean trees and application to the Catalan satisfiability problem. In Springer-Verlag, editor, *Latin American Theoretical INformatics*, volume 8392, pages 466–477, 2014.

[22] O. Bodini and A. Genitrini. Cuts in increasing trees. In *12th SIAM Meeting on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 66–77, 2015.

[23] O. Bodini, A. Genitrini, F. Peschanski, and N. Rolin. Associativity for Binary Parallel Processes : A Quantitative Study. In *Algorithms and Discrete Applied Mathematics - First International Conference, CALDAM*, pages 217–228, 2015.

[24] O. Bodini, M. Dien, X. Fontaine, A. Genitrini, and H.-K. Hwang. Increasing Diamonds. In *12th Latin American Symposium (LATIN)*, pages 207–219, 2016.

[25] A. Genitrini. Full asymptotic expansion for Pólya structures. In *27th International Meeting on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AofA)*, pages 152–163, 2016.

[26] O. Bodini, M. Dien, A. Genitrini, and F. Peschanski. The ordered and colored products in analytic combinatorics : Application to the quantitative study of synchronizations in concurrent processes. In *14th SIAM Meeting on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 16–30, 2017.

[27] O. Bodini, M. Dien, A. Genitrini, and F. Peschanski. Entropic uniform sampling of linear extensions in series-parallel posets. In *12th International Computer Science Symposium in Russia (CSR)*, pages 71–84, 2017.

Autres

[28] A. Genitrini. *Expressions Booleénnes aléatoires : probabilité, complexité et comparaison quantitative de logiques propositionnelles*. Thèse de doctorat, Université de Versailles-Saint-Quentin-en-Yvelines, 2009.

[29] A. Genitrini, B. Gittenberger, M. Kauers, and M. Wallner. Asymptotic enumeration of compacted binary trees. Rapport technique, http ://arxiv.org/abs/1703.10031, 2017.

Notons que seuls les articles [6, 25] ne sont pas cités dans le mémoire, pour la simple raison que les études qu'ils présentent sont éloignées de la logique et de la concurrence.

Table des matières

Première partie

Logique quantitative : Arbres booléens et fonctions booléennes aléatoires

Chapitre 1

Introduction et préliminaires

Paris et al. [PVW94], puis Lefman et Savický [LS97] ont cherché à définir une distribution de probabilité *naturelle* sur les fonctions booléennes, induite par des expressions construites avec les deux connecteurs `et` et `ou` et un nombre k fixé de variables (et leurs négations) – ce système est complet, i.e. toute fonction booléenne sur k variables est représentable par des expressions. L'approche de Lefman et Savický se base sur des arbres de taille infinie qui sont élagués suivant certaines contraintes. En 2004, Chauvin et al. [CFGG04] présentent une autre approche fondée sur une suite de proportions d'arbres de taille fixée et démontrent que la limite de cette suite définit bien une distribution de probabilité sur les fonctions booléennes, qui par ailleurs est identique à celle de Lefman et Savický.

L'article de synthèse de Gardy [Gar06] expose des résultats numériques pour un nombre restreint de variables. La distribution de probabilité est clairement biaisée vers les fonctions qui s'expriment le plus simplement, c'est-à-dire dont il existe des petits arbres les représentant. En raison du grand nombre de fonctions booléennes à k variables, 2^{2^k}, on se trouve très rapidement dans l'impossibilité d'obtenir des résultats numériques lorsque k augmente.

L'article [Gar06] présente aussi les outils d'analyse combinatoire permettant de s'intéresser au cas où le nombre de variables k tend vers l'infini. Ces outils classiques en combinatoire analytique sont très largement présentés et étudiés dans le livre de Flajolet et Sedgewick [FS09].

Enfin, les articles [LS97], [CFGG04] et [Koz08] présentent des bornes sur la probabilité d'une fonction donnée, dépendant directement de sa *complexité* (i.e. la taille des arbres les plus petits la représentant). Là aussi les résultats semblent biaisés vers les fonctions de petite complexité, mais ne donnent que peu d'information sur les fonctions de grande complexité. Ces articles exhibent des bornes sur les probabilités de fonctions mais qui ne permettent pas d'exhiber de sous-famille (non triviale) de fonctions dont la probabilité est strictement positive lorsque le nombre de variables tend vers l'infini.

Voilà quelques questions auxquelles nous souhaiterions répondre. *La distribution sur les fonctions booléennes, est-elle biaisée vers les fonctions de petites complexités en raison du système de connecteurs utilisés, ou le résultat est-il plus fortement lié à la structure d'arbre sous-jacente ? L'effet Shannon, qui dit, informellement, que la plupart des fonctions sont de grande complexité (pour la distribution uniforme des fonctions), a-t-il aussi lieu pour ces distributions de probabilités induites par des expressions arborescentes ? Qu'en est-il de la complexité moyenne des fonctions suivant cette distribution de probabilité ?*

Afin de construire une expression booléenne, nous avons à notre disposition deux ensembles de symboles :

— $\mathcal{V}$ un ensemble de littéraux, contenant éventuellement, aussi les constantes `vrai` ou `faux` ;

— $\mathcal{C}$ un ensemble de connecteurs.

Nous appellerons par la suite cet ensemble $\{\mathcal{V}, \mathcal{C}\}$ *système logique* (ou simplement système). Notons que l'ensemble $\mathcal{V}$ peut contenir des littéraux positifs (les variables) et des littéraux négatifs (les négations de variables). Par ailleurs, à chaque connecteur de $\mathcal{C}$ est associée une arité [1]. Par exemple, le `ou` logique est (généralement) d'arité 2, alors que la négation est d'arité 1. Enfin nous avons besoin de parenthèses afin de combiner entre elles plusieurs sous-expressions.

Définition 1.0.1

Une expression booléenne est soit un atome de $\mathcal{V}$, soit construite à partir d'un connecteur de $\mathcal{C}$, d'arité α et d'une suite de α expressions booléennes qui sont les arguments du connecteur.

Par exemple, $(x_2 \vee ((\bar{x}_1 \to x_3) \vee x_7)$ est un expression booléenne, pour laquelle $\vee$ correspond au connecteur logique `ou`, $\to$ est l'`implication`, x_2, x_3 et x_7 sont des variables (littéraux positifs) et $\bar{x}_1$ est le littéral négatif associé à la variable x_1. Nous remarquons qu'à chaque expression booléenne nous pouvons associer naturellement une représentation arborescente. La Figure 1.1 est la représentation arborescente de l'expression précédente.

Pour une expression booléenne A, sa *taille*, notée $|A|$, est le nombre d'occurrences de littéraux qu'elle contient. Ainsi, dans sa représentation arborescente, la taille correspond au nombre des feuilles de l'arbre.

1. L'arité d'un connecteur correspond au nombre d'arguments nécessaires au connecteur, appelé aussi degré sortant.

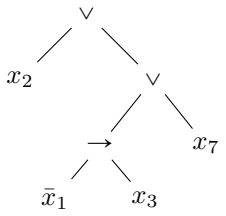


FIGURE 1.1 – *La représentation arborescente de l'expression* $(x_2 \vee ((\bar{x}_1 \to x_3) \vee x_7)$.

Étant donné une expression et une évaluation de chacune des variables, l'expression s'évalue à l'une des deux constantes `vrai` ou `faux`, via les règles sémantiques des connecteurs logiques. La représentation, en logique classique, des règles d'un connecteur peut se faire à l'aide de sa table de vérité.

Définition 1.0.2

Étant donné une expression, elle est appelée tautologie *si pour chacune des évaluations des variables, l'expression s'évalue à* `vrai`.

En prenant un peu de recul, nous remarquons qu'une tautologie est une représentation d'un *théorème*. Par exemple, si l'expression $A \to B$ est une tautologie, cela peut s'interpréter ainsi : en prenant la sous-expression A en tant qu'hypothèse, nous pouvons démontrer que la sous-expression B est vraie.

À chaque expression booléenne est associée une *fonction booléenne*. Une fonction (sur k variables) est une application de l'ensemble $\{\texttt{vrai}, \texttt{faux}\}^k$ dans l'ensemble des deux booléens `vrai` et `faux`. En fixant le nombre de variables à k, il existe 2^{2^k} fonctions. Mais en laissant augmenter la taille des expressions (construites avec uniquement k variables) vers l'infini, le nombre d'expressions n'est pas borné. Ainsi, pour chaque fonction, représentable dans un système donné, il existe de multiples expressions la représentant. Par exemple, $x_1 \to x_2$ et $(x_1 \vee x_1) \to x_2$ représentent la même fonction booléenne. Une façon aisée de le prouver consiste à montrer que les tables de vérité associées à chacune des deux expressions est la même. Par la suite, nous dirons qu'une expression *calcule* une fonction booléenne. Ainsi l'expression $x_1 \vee \bar{x}_1$, calcule la fonction constante `vrai`, l'expression $x_1 \vee (x_2 \wedge \bar{x}_2)$, où $\wedge$ correspond au connecteur logique `et`, calcule la fonction projection $(x_1, x_2, \ldots, x_k) \mapsto x_1$ (cette représentation de la fonction est abusive, mais classique).

Pour une fonction f expressible dans un système logique, nous appelons *complexité* de la fonction, et notons $L(f)$, la taille de la plus petite expression du système la calculant. Les expressions calculant f et de taille $L(f)$ sont appelées *expressions minimales* ou *arbres minimaux* de f, et l'ensemble de arbres minimaux est noté $\mathcal{M}_f$. Notons que dans deux systèmes différents la même fonction n'a pas forcément la même complexité. En effet, dans le système construit avec l'unique connecteur $\to$ et les littéraux positifs, un arbre minimal pour la fonction $(x_1, x_2, \ldots, x_k) \mapsto x_1 \vee x_2$ est $(x_1 \to x_2) \to x_2$ (taille 3), alors que dans les systèmes contenant le connecteur $\vee$ et les littéraux positifs, nous construisons aisément une expression de taille 2 calculant la même fonction. En outre, il se peut que certaines fonctions ne soient pas représentables au sein d'un système donné. Par exemple, la fonction constante `faux` ne peut pas être représentée dans le système construit avec l'unique connecteur $\to$ et les littéraux positifs.

Un système logique étant fixé, nous nous intéressons aux structures arborescentes des expressions constructibles. Ainsi, à taille n fixée, nous pouvons définir la proportion d'un ensemble d'arbres vérifiant certaines propriétés parmi tous les arbres de cette taille. Par exemple, pour une fonction booléenne fixée, nous pouvons, calculer la proportion d'arbres de taille n la représentant. La limite de la suite de proportions (lorsque n augmente), permet dans certains cas, de définir la probabilité de la fonction booléenne en question. Afin qu'un système logique définisse une distribution de probabilité sur les fonctions booléennes, il suffit qu'il vérifie des conditions provenant des théorèmes de Drmota [Drm97], Lalley [Lal93] et Woods [Woo97], qui sont trois versions plus ou moins riches du même résultat. Le résultat global est décrit de façon globale [FS09, p. 489]. Dans les articles [2, 9], par exemple, le lecteur trouvera une application de ces théorèmes dans les systèmes contenant soit le connecteur `implication` soit les connecteurs `et, ou` (avec une notion de taille particulière pour ce dernier système).

Définition 1.0.3

Soit $\mathcal{F}_k$ l'ensemble des arbres du système (sur k variables). La fraction limite *d'un sous-ensemble $\mathcal{A}$ d'arbres est définie par*

$$\mu_k(\mathcal{A}) = \lim_{n\to\infty} \frac{|\{A \in \mathcal{A}, |A| = n\}|}{|\{A \in \mathcal{F}_k, |A| = n\}|},$$

lorsque cette limite existe. Notons $\mathcal{F}_k(f)$ l'ensemble des arbres calculant une fonction f ; dès lors nous définissons la probabilité $\mu_k(f)$ de la fonction f *par la fraction limite de l'ensemble $\mathcal{F}_k(f)$.*

Notons par exemple que, pour chaque famille d'arbres représentant une fonction booléenne, l'existence de la limite de la fraction découle des théorèmes de Drmota, Lalley et Woods. Néanmoins pour certaines familles d'arbres (cf. par exemple les tautologies intuitionnistes du Chapitre 2), nous n'avons pas établi l'existence de la limite. Dès lors, nous nous intéressons aux limites inférieure et supérieure et nous utilisons $\mu_k(\cdot)$ lorsque le résultat annoncé est vrai aussi bien pour la limite inférieure que pour la limite supérieure (cf. l'article [3]).

La suite de cette partie présente un chapitre pour chaque problème qui nous a intéressé dans le contexte des arbres booléens et fonctions booléennes associées.

Chapitre 2

Logique intuitionniste

L'*intuitionnisme* consiste originellement en une position philosophique par rapport aux mathématiques. Ses fondements, en tant que branche de la logique formelle, ont été définis par Brouwer dans les années 1930. Certains raisonnements, jugés contre-intuitifs, sont rejetés dans cette logique. En effet, une règle du calcul propositionnel, définie en logique classique, n'est pas admise en logique intuitionniste : il s'agit du *tiers exclu* qui consiste à dire qu'étant donné une proposition ϕ, soit *"ϕ est* `vrai` *"*, soit *"non(ϕ) est* `vrai` *"*. Cette règle peut se formaliser en : *l'expression $\phi \vee \overline{\phi}$ est une tautologie.* Le lecteur trouvera l'histoire et des motivations du développement de cette logique présentées par Van Dalen [vD86] ou dans le premier chapitre du livre [TD88].

La logique intuitionniste se veut *constructive*. Ainsi on ne veut pas seulement connaître l'existence d'une solution pour un problème, mais on veut aussi savoir la construire. Voilà un exemple de théorème : *Il existe des nombres irrationnels a et b tels que a^b est un nombre rationnel.* Présentons-en une preuve en logique classique : *si $\sqrt{2}^{\sqrt{2}}$ est rationnel, alors $a = b = \sqrt{2}$ conviennent. Sinon, prenons $a = \sqrt{2}^{\sqrt{2}}$ et $b = \sqrt{2}$. Dès lors $a^b = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = \sqrt{2}^2 = 2$ est un nombre rationnel.* Cette démonstration est rejetée par les intuitionnistes du fait que la solution n'est pas exhibée. Elle utilise le tiers exclu et par conséquent on ne connaît pas les valeurs de a et de b qui satisfont la propriété. Remarquons néanmoins, pour cet exemple, il existe une preuve constructive pour montrer que $\sqrt{2}^{\sqrt{2}}$ est irrationnel, donc cette proposition est aussi un théorème pour la logique intuitionniste. Toutefois, pour certaines propositions, nous ne sommes pas en mesure d'exhiber une preuve intuitionniste. Pour les logiciens intuitionnistes, savoir qu'une proposition ou son contraire est vraie n'apporte pas suffisamment d'information. Soit la proposition suivante : *dans la suite des décimales de π, il existe un rang où les dix chiffres apparaissent de manière consécutive et croissante.* Parmi les 200 premiers millions de décimales cette proposition est fausse, mais dans la suite des décimales, il est possible que jamais personne ne sache dire si cette proposition est vraie ou fausse.

Afin de montrer que l'expression booléenne $x \to y$ est équivalente à l'expression $\bar{x} \vee y$ il est nécessaire d'avoir recours à la règle du tiers exclu. Ainsi, la logique intuitionniste ne définit plus le concept d'*implication*, de la manière précédente. En affirmant $x \to y$, on veut désormais un moyen de *construction* d'une démonstration de y à partir d'une démonstration de x. Enfin, terminons sur le fait que les démonstrations en logique intuitionniste (i.e. les tautologies intuitionnistes) sont aussi des démonstrations en logique classique (i.e. tautologies classiques). Le lecteur trouvera des informations sur la logique, et la logique intuitionniste en particulier, dans le livre de David et al. [DNR04], par exemple. Voilà deux tautologies valides en logique classique mais non valides en logique intuitionniste.

$$\begin{array}{rcl} x \vee \bar{x} & : & \text{tiers exclu} \\ \bar{\bar{x}} \to x & : & \text{double négation.} \end{array}$$

Les travaux du 20-ème siècle, notamment la correspondance de Curry-Howard (voir le livre de Sørensen et Urzyczyn [SU98] par exemple), ont donné à la logique intuitionniste un statut central en logique et en informatique, en en faisant historiquement la première des logiques constructives. C'est précisément la correspondance de Curry-Howard qui montre qu'une preuve constructive a toutes les propriétés d'un programme ; i.e. que les règles de démonstration permettent d'écrire des programmes corrects.

Afin de distinguer les tautologies intuitionnistes des tautologies non-intuitionnistes (l'union des deux ensembles est habituellement appelée tautologies classiques), nous proposons au lecteur de se référer aux travaux dans le contexte de l'algèbre de Heyting, notamment détaillés dans le livre [SU98], et explicités, dans notre contexte de logique quantitative dans les articles [14] et [3]. Dotés de cette formalisation, nous sommes en mesure de vérifier si une tautologie est intuitionniste ou non.

Dès le début des années 2000, des logiciens de Cracovie ont pour but de comparer l'expressivité de la logique intuitionniste au sein de la logique classique. Mentionnons en particulier l'article [MTZ00] dont le but est de calculer la proportion de types inhabités au sein des types du premier ordres (à taille fixée). Via la correspondance de Curry-Howard, cette proportion correspond à celle des tautologies intuitionnistes au sein de l'ensemble des tautologies (classiques), toujours à taille fixée.

L'approche classique développée en combinatoire analytique dans le contexte d'études quantitatives dans ce

contexte, s'est rapidement montrée être la clé pour résoudre des problèmes de ce type, dans ce contexte. En effet, un résultat central montre que les expressions booléennes vérifiant certaines propriétés sont pour la plupart construites de la manière la plus simple possible. Par conséquent, il est aisé d'en donner une spécification constructive non ambigüe, puis la comparaison quantitative par rapport à d'autres classes d'expressions se fait de manière usuelle.

Comparer les logiques intuitionniste et classique correspond à quantifier la proportion de tautologies classiques qui sont également des tautologies intuitionnistes.

2.1 Système logique de l'`implication`

Au sein de la logique propositionnelle (complète), un système, simple dans le sens qu'il ne comporte qu'un unique connecteur, l'`implication` et seulement les littéraux positifs, permet déjà de distinguer la logique intuitionniste de la classique. En effet, Peirce [Pei85] a présenté l'expression suivante, désormais appelée la loi de Peirce :

$$((x \to y) \to x) \to x. \tag{2.1}$$

La preuve du fait que cette expression est une tautologie (classique) nécessite le recours au tiers-exclu, ou, de façon équivalente un raisonnement par l'absurde. Il n'existe pas de preuve constructive de cette tautologie. Ce n'est pas une tautologie intuitionniste. Nous nous sommes intéressés à ce système de l'`implication` dans les articles [11, 14] et [1].

Quelle est la proportion de tautologies intuitionnistes dans ce système minimaliste de connecteurs et littéraux, respectivement $\{\to\}$ et $\{x_1, x_2, \dots, x_k\}$?

Commençons par définir une spécification de la classe $\mathcal{A}$ des expressions constructibles.

$$\mathcal{A} = x_1 \mid x_2 \mid \cdots \mid x_k \mid \mathcal{A} \to \mathcal{A}.$$

On représente aisément les expressions sous forme arborescente, dont la structure est un arbre binaire où les nœuds internes (binaires) sont décorés avec le connecteur $\to$ et les feuilles contiennent une variable. Dans ce chapitre, la *taille* de l'arbre est son *nombre de feuilles*. La structure des expressions est suffisamment classique pour qu'on observe immédiatement que le nombre d'expressions de taille n vaut

$$\mathrm{Cat}_{n-1}\, k^n = \frac{1}{n}\binom{2n-2}{n-1} k^n, \tag{2.2}$$

où Cat_{n-1} est un nombre de Catalan, plus exactement, le nombre d'arbres binaires à $n-1$ nœuds internes.

On s'intéresse désormais, d'un point de vue quantitatif, à l'étude de sous-classes particulières d'expressions, par exemple les tautologies qui sont ou non intuitionnistes. L'idée fondamentale consiste à partitionner l'ensemble des expressions booléennes en les trois familles suivantes : les expressions qui ne sont pas des tautologies, les tautologies intuitionnistes et les tautologies classiques mais non intuitionnistes. Néanmoins, la spécification (par décomposition) de l'ensemble des expressions rentrant dans chacune de ces familles ne semble pas atteignable. Par conséquent, afin d'obtenir des résultats quantitatifs, nous construisons des sous-familles de ces ensembles, suffisamment riches pour que, asymptotiquement quand la taille des expressions considérées tend vers l'infini, on réussisse à prendre en compte l'essentiel des expressions.

Dans le système construit avec le connecteur de l'`implication`, du fait de la dissymétrie du connecteur (relativement à sa sémantique), il s'avère efficace de décomposer les expressions selon leur branche droite. Ainsi, nous définissons les deux notions suivantes : pour une expression

$$A_1 \to (A_2 \to (\cdots \to (A_p \to x)\dots)),$$

les sous-expressions $A_1, A_2, \dots A_p$ sont appelées *prémisses* et la variable x (étiquetant la feuille la plus à droite de la représentation arborescente) est le *but* de l'expression.

Via cette décomposition, nous définissons une famille importante de tautologies.

Définition 2.1.1

L'ensemble des tautologies simples *contient les expressions dont l'une des prémisses est réduite à une feuille étiquetée avec la même variable que le but de l'expression.*

Proposition 2.1.2

Les tautologies simples sont des tautologies intuitionnistes.

L'intuition fondamentale aboutissant à ce résultat réside dans le fait que l'on peut voir les prémisses d'une expression comme des hypothèses et que le but de l'expression est ce qui est à démontrer. Si on prend en hypothèse, le résultat à démontrer, alors l'expression est évidemment toujours vraie.

Théorème 2.1.3

Dans le système composé de l'`implication` et des littéraux positifs $\{x_1, \dots x_k\}$, on a les fractions limite suivantes

$$\mu_k(\textit{Expr. non tauto.}) \underset{k\to\infty}{=} 1 - \frac{1}{k} - \frac{5}{4k^2} + O\left(\frac{1}{k^3}\right);$$

$$\mu_k(\textit{Tauto. intuitionnites}) \underset{k\to\infty}{=} \frac{1}{k} + \frac{3}{4k^2} + O\left(\frac{1}{k^3}\right);$$

$$\mu_k(\textit{Tauto. non intuit.}) \underset{k\to\infty}{=} \frac{1}{2k^2} + O\left(\frac{1}{k^3}\right).$$

On rappelle que les tautologies classiques sont partitionnées entre les tautologies intuitionnistes et celles qui ne

sont pas intuitionnistes. Par conséquent, en première approximation, asymptotiquement, lorsque la taille des expressions tend vers l'infini, et que le nombre de variables k tend vers l'infini, les tautologies classiques sont presque sûrement des tautologies intuitionnistes. Ce que l'on peut synthétiser sous la forme du corollaire suivant :

Corollaire 2.1.4

Dans le système composé de l'`implication` *et des littéraux positifs* $\{x_1, \ldots, x_k\}$,

$$\lim_{k\to\infty} \frac{\mu_k(\textit{Tauto. intuitionnites})}{\mu_k(\textit{Tauto. classiques})} = 1.$$

Démonstration (idées) : On remarque que la plus petite tautologie (intuitionniste) s'écrit $x \to x$. En particulier, elle ne nécessite qu'une seule répétition d'un variable donnée. La loi de Peirce (2.1), quant à elle, nécessite 3 occurrences de la même variable. Au sein des résultats précédents, cette notion d'occurrences multiples est la clé des preuves. Intuitivement, chaque occurrence, nécessaire, d'une variable fixée fait décroître la fraction limite d'un facteur $1/k$ pour les familles d'expressions. Nous remarquons d'ailleurs que nous construisons aisément une famille infinie d'expressions autour de la loi de Peirce, en remplaçant la variable y par une sous-expression quelconque (ou presque : si l'on veut conserver le fait que la tautologie reste non intuitionniste, il y a quelques contraintes supplémentaires à respecter). ■

Ces premiers résultats établis, une question fondamentale se pose. Lors du calcul de la fraction limite, deux passages successifs à la limite sont effectués : la proportion des expressions dont la taille tend vers l'infini est calculée, puis afin d'appréhender le résultat, le nombre de variables se rapproche de l'infini. Bien entendu, il ne semble pas immédiat que l'échange des deux limites donne le même résultat. Kozik, Zaionc et moi-même avons alors modélisé (cf. l'article [12]) les expressions différemment afin d'autoriser un nombre illimité de variables lors de la construction des expressions. Toutefois, pour rester dans le contexte de la combinatoire analytique, plus précisément pour que les familles respectent les contraintes des classes combinatoires, nous introduisons une relation d'équivalence telle que deux expressions sont équivalentes si elles sont identiques à un renommage (par α-équivalence) des variables. Dès lors, le nombre d'expressions de taille n (cf. Équation (2.2)) devient

$$\mathrm{Cat}_{n-1} \cdot B_n,$$

où B_n est le n-ième nombre de Bell, c'est-à-dire le nombre de manières de partitionner la suite des n feuilles des arbres (suite obtenue via un parcours arbitraire de l'arbre). Dans ce contexte, l'adaptation du corollaire précédent donne :

Corollaire 2.1.5

Dans le système composé l'`implication` *et d'une infinité de littéraux positifs, dans lequel les expressions sont équivalentes à renommage près, nous avons*

$$\lim_{k\to\infty} \frac{\mu_k(\textit{Tauto. intuitionnites})}{\mu_k(\textit{Tauto. classiques})} = 1.$$

Démonstration (idées) : Le résultat est directement obtenu par adaptation des familles d'expressions établie dans le cadre du premier modèle. ■

Plus que le résultat en lui-même, ce qu'il faut retenir c'est que l'approche que nous établissons au sein des deux modèles est robuste et permet d'obtenir des résultats similaires.

2.2 Logique propositionnelle complète

Dans un souci d'extension de ces résultats, nous nous sommes intéressés (cf. [15, 3]) à un système logique central pour la logique dite propositionnelle. Dans ce but, d'un point de vue syntaxique, les connecteurs utilisés définissent l'ensemble $\mathcal{C} = \{\to, \wedge, \vee\}$ et les littéraux, $\mathcal{V} = \{x_1, x_2, \ldots, x_k, \bot\}$, le dernier symbole consiste en la constante `faux`. Il est bien connu que toute fonction booléenne peut s'exprimer avec uniquement les deux connecteurs $\wedge$ et $\vee$ en s'appuyant sur les littéraux positifs et négatifs : ce type de système est dit *complet*. Dans notre contexte, nous n'utilisons que les littéraux positifs, néanmoins, nous remarquons que les littéraux négatifs sont aisément constructibles : $\bar{x} \equiv x \to \bot$. Par conséquent, ce système est également complet.

Dans ce nouveau contexte, toujours en exhibant des sous-familles d'expressions construites sur des motifs d'expressions les plus simples, nous prouvons le théorème suivant.

Théorème 2.2.1

Dans le système composé des connecteurs $\{\to, \wedge, \vee\}$ *et des littéraux,* $\{x_1, x_2, \ldots, x_k, \bot\}$, *nous avons*

$$\lim_{k\to\infty} \frac{\mu_k(\textit{Tauto. intuitionnites})}{\mu_k(\textit{Tauto. classiques})} = \frac{5}{8}.$$

Bien que les résultats soient différents de ceux obtenus dans le système basé uniquement sur l'`implication`, les méthodes de preuves sont robustes et s'adaptent relativement bien.

Nous observons, dans ce contexte, que le comportement asymptotique des fractions limite des tautologies intuitionnistes et des non intuitionnistes sont du même ordre ; par conséquent, la proportion des tautologies intuitionnistes (parmi les tautologies) ne tend plus vers la constante 1. Dans ma thèse [28, chap. 9], je me suis intéressé à toutes les variantes concernant l'ensemble des connecteurs utilisés. Les preuves sont robustes et seuls des détails calculatoires font varier les résultats.

Enfin, en adaptant les modèles au contexte d'un ensemble de variables non borné, comme dans le cas d'un unique connecteur, l'`implication`, les fonctions génératrices font intervenir les nombres de Bell, mais les résultats finaux sont identiques au Théorème 2.2.1 (où seul le contenu de l'ensemble $\mathcal{V}$ change).

Chapitre 3

Distributions de probabilité des fonctions booléennes

Ce chapitre met en avant la démarche que nous avons suivie durant les dix dernières années afin d'enrichir au fur et à mesure nos modèles et d'analyser l'évolution des distributions induites sur les fonctions booléennes. Le résultat sur lequel nous nous concentrons ici consiste à fixer une fonction et à étudier le premier ordre du comportement asymptotique de sa probabilité au sein des trois modèles (principaux) suivants :

- $\{\rightarrow\}$; $\{x_1, \dots x_k\}$; taille = nb. feuilles
- $\{\wedge, \vee\}$ avec règles d'associativité ou de commutativité ; $\{x_1, \bar{x}_1 \dots x_k, \bar{x}_k\}$; taille = nb. feuilles
- $\{\wedge, \vee\}$ avec règle d'associativité ; $\{x_1, \bar{x}_1 \dots x_k, \bar{x}_k\}$; taille = nb. nœuds.

Bien que les aspects techniques des preuves soient essentiellement différents, nous avons été surpris que des résultats très similaires soient valides pour chacun des trois systèmes logiques.

3.1 Modèles originels

Nous reprenons le système logique construit sur l'`implication` et les littéraux positifs. Dans le chapitre précédent nous nous sommes concentrés sur une petite famille d'expressions qui calculent la fonction constante `vrai`. L'ensemble complémentaire de ces expressions peut naturellement être partitionné suivant les fonctions booléennes calculées par les expressions. Les résultats dans ce sens sont issus de nos articles [13] et [2]. Nous avions rappelé que le système basé sur l'`implication` et uniquement k littéraux positifs n'est pas complet. Par exemple, la fonction constante `faux` n'est calculée par aucune expression. De façon plus précise, si l'on note $\mathfrak{F}_k$ l'ensemble (de taille 2^{2^k}) des fonctions booléennes construites sur les k variables $x_1, \dots, x_k$, alors une fonction f admet des expressions la calculant dans ce système logique s'il existe une variable $x \in \{x_1, \dots, x_k\}$ et une fonction (quelconque) $g \in \mathfrak{F}_k$ telles que $f = x \vee g$.

Prenons pour exemple pour la suite de cette section la fonction $\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee x_4$. La complexité de cette fonction est 5. En effet, il n'existe pas d'arbre de taille 4 qui la calcule, et l'arbre représenté sur la Figure 3.1 en est un des arbres minimaux.

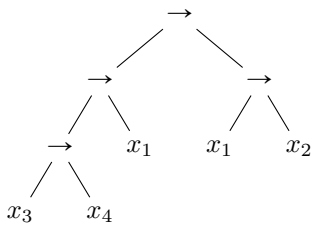


FIGURE 3.1 – *Un arbre minimal pour la fonction* $f = \bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee x_4$.

Théorème 3.1.1

Dans le système composé de l'`implication` *et des littéraux positifs* $\{x_1, \dots x_k\}$*, étant donné une fonction* f*, expressible et de complexité* $L(f)$*, sa probabilité vaut*

$$\mu_k(f) \underset{k\to\infty}{=} \frac{\lambda(f)}{4^{L(f)}\, k^{L(f)+1}} + O\left(\frac{1}{k^{L(f)+2}}\right),$$

où $\lambda(f)$ *est une valeur ne dépendant que de* f *(et pas de* k*).*

Démonstration (idées) : Étant donné une fonction f expressible et qui dépend d'un nombre fixé de variables, on cherche une famille d'expressions simples qui la calculent. Puis nous montrons que, asymptotiquement, au premier ordre, essentiellement toutes les expressions représentant f appartiennent à cette famille. La famille d'expressions simples pour f est construite à partir des arbres minimaux de la fonction. Tout d'abord étant donné un arbre minimal M (de taille $L(f)$, indépendant de k et n), nous pouvons remplacer arbitrairement l'un de ses sous-arbres S par $\tilde{S} := T \rightarrow S$ où T est une tautologie. On remarque que les tables de vérité associées aux sous-arbre S et $\tilde{S}$ sont identiques. Par conséquent, le nouvel arbre $\tilde{M}$ issu de la substitution de S par $\tilde{S}$ dans M calcule la même fonction f. On dit qu'on a effectué une *expansion de type tautologie* dans l'arbre minimal. Du fait que la fraction limite des tautologies est de l'ordre de k^{-1}, on montre aisément que la fraction limite de cette famille d'arbres calculant f se comporte en $\Theta(k^{-L(f)-1})$. Dans la figure suivante, nous avons représenté l'arbre minimal de la Figure 3.2 dans lequel nous avons effectué une expansion avec une tautologie simple. Ce nouvel arbre calcule

toujours la même fonction que l'arbre minimal originel.

Afin d'obtenir une famille d'expressions calculant f dont l'équivalent asymptotique coïncide avec celui de f, au premier ordre, il nous suffit d'introduire deux autres types d'expansions, les expansions de type *but* et de type *prémisse*, à peine plus compliquées que l'expansion de type tautologie. Ces expansions consistent à contraindre le but ou l'une des prémisses de la sous-expression que l'on ajoute en tant qu'expansion. Dans la partie droite de la Figure 3.2, on a représenté deux sous-arbres, qui, en remplaçant, dans l'arbre ci-contre, le sous-arbre (de couleur bleue) qui est une tautologie simple, donneraient une expansion de type but x_3 pour l'une et de type prémisse x_4 pour l'autre. Finalement, en se basant sur ces trois types d'expansions dans les arbres minimaux de f, on obtient une borne inférieure de la famille des expressions de f dont le premier ordre est suffisant. En effet, pour conclure la preuve nous construisons une famille, incluant toutes les expressions calculant f, et dont nous sommes en mesure de calculer la fraction limite dont le premier ordre coïncide avec celui de la borne inférieure. ■

On remarque que pour chacun des trois types d'expansion, une seule étiquette de feuille est contrainte, et par conséquent les familles d'arbres de telles expansions ont pour une fraction limite égale à $\Theta(k^{-1})$. Par ailleurs, notons que la valeur $\lambda(f)$ exhibée dans le Théorème 3.1.1 correspond au nombre d'expansions que l'on peut faire dans l'ensemble des arbres minimaux.

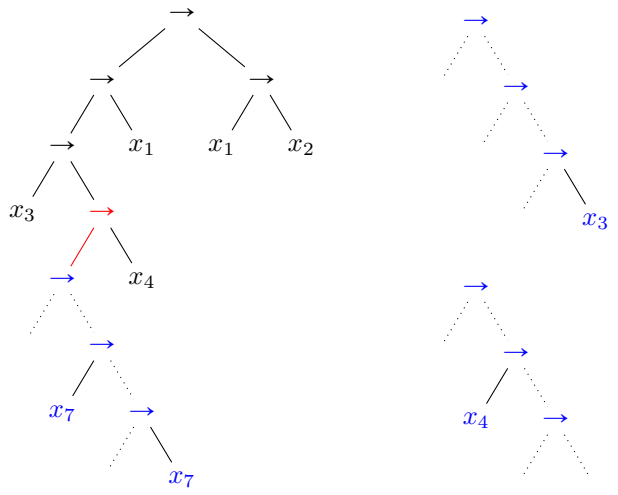


FIGURE 3.2 – *(à gauche) Une expansion, de type tautologie, d'un arbre minimal de la fonction* $f = \bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee x_4$ *; (à droite) Deux expansions de type* but *ou* prémisse*, qui peuvent être substituées à l'expansion tautologie ci-contre*

En parallèle de nos travaux dans le système logique de l'`implication`, Kozik [Koz08] a démontré des résultats similaires dans le système basé sur les connecteurs `et` et `ou` et les littéraux positifs et négatifs. L'approche qu'il a utilisée est analogue à la nôtre. En particulier, les deux modèles arborescents vérifient les mêmes caractéristiques et il semble naturel désormais, avec le recul d'obtenir des résultats similaires. Néanmoins, il a formalisé son approche dans un contexte relativement générique, sous le nom de *langage de motifs*. Ce traitement permet de simplifier la partie technique et calculatoire nécessaire à la justification de l'équivalent asymptotique de la probabilité d'une fonction fixée.

3.2 Commutativité ou l'associativité

En se basant sur ces deux travaux, nous voulons ajouter les règles de commutativité et d'associativité classiques pour ces connecteurs logiques. Nous avons ainsi considéré, dans l'article [5], la commutativité et l'associativité directement au sein des expressions booléennes. La première règle transforme les arbres en version non plane et par conséquent fait intervenir les résultats de Pólya [Pó37] concernant la théorie des *indices de cycle*. De son côté, la règle d'associativité ajoute des contraintes dans l'agencement des connecteurs au sein des expressions. Il est à noter que les résultats sont très similaires à ceux obtenus lors des premiers travaux. Remarquons toutefois que la non-planarité ou les contraintes de composition entre connecteurs ajoutent des difficultés techniques dans les preuves. Le point central de l'ensemble de ces modèles est le fait que la taille des expressions correspond toujours au nombre de feuilles des expressions (même dans le modèle associatif, où les expressions ne sont plus binaires et donc pour lesquelles il n'y a plus de relation entre les nombres de feuilles et de nœuds internes). La Figure 3.4 exhibe un tel exemple.

Dans le modèle de l'implication (cf. l'article [4]), nous pouvons, en quelque sorte, définir une règle proche de la commutativité : nous remarquons en effet que deux expressions, identiques, à permutation près des prémisses (et récursivement dans les sous-expressions), calculent la même expression. Par exemple,

$$\begin{aligned} &A_1 \to (A_2 \to (\cdots \to (A_p \to x)\dots)) \\ &\quad \equiv A_{\sigma(1)} \to (A_{\sigma(2)} \to (\cdots \to (A_{\sigma(p)} \to x)\dots)), \end{aligned}$$

où σ est une permutation de $1, \dots, p$ et $A \equiv A'$ correspond au fait que A et A' calculent la même fonction booléenne. Ainsi, en définissant les expressions booléennes à permutation près des prémisses, nous pouvons construire un modèle arborescent d'expressions isomorphe aux expressions suivantes : la racine de l'expression est étiquetée avec le but de l'expression et est reliée à chacun des éléments qui forment l'ensemble de prémisses. Par exemple, l'arbre de la Figure 3.1 est réencodé dans ce modèle dans la Figure 3.3, où les enfants de chaque nœud ne sont pas ordonnés. Notons que dans cet exemple (Figure 3.3), les nœuds internes sont tous binaires, puisque l'expression et ses sous-expressions contiennent chacune deux prémisses (ou aucune). Les nœuds peuvent néanmoins être d'arité quelconque. Naturellement, la notion de taille pour ce modèle est le nombre d'occurrence des variables, i.e. le nombre de nœuds de l'arbre (ce qui correspond aussi au nombre de feuilles dans la représentation originelle).

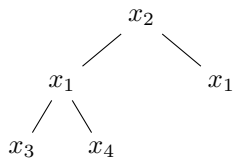


FIGURE 3.3 – *Représentation des expressions à permutation des prémisses près.*

Dans le contexte d'un système logique `et/ou`, introduire la règle d'associativité se traduit par des connecteurs d'arité quelconque et une alternance des connecteurs sur chaque chemin de la racine vers une feuille de l'arbre. On dira que ces arbres sont stratifiés. Cette représentation se reflète sur des expressions arborescentes moins profondes que les expressions originelles (cf. par exemple la Figure 3.4).

La règle d'associativité telle que nous l'utilisons consiste à compresser *autant que possible* la structure interne des expressions comme sur la Figure 3.4. Dans ce modèle, une notion de taille naturelle correspond à la taille mémoire de stockage nécessaire. Donc une première approximation consiste à prendre le nombre de nœuds de l'arbre comme notion de taille.

Par ailleurs, considérons le système logique des expressions construites avec les connecteurs `et` et `ou`, les littéraux positifs et négatifs et la règle d'associativité. Sur

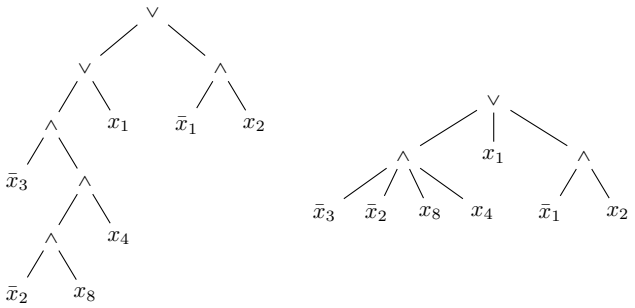


FIGURE 3.4 – *(à gauche) Arbre* `et/ou` *classique ; (à droite) L'arbre ci-contre avec la règle d'associativité*

la Figure 3.4, nous remarquons que les suites de connecteurs identiques et consécutifs se rétractent en un seul connecteur. Nous pouvons également ajouter la règle de commutativité dans ce dernier modèle, ce qui entraîne qu'un nœud possède un ensemble, plutôt qu'une suite, de descendants.

Dans ces différents modèles avec ou sans associativité et commutativité, dans lesquels nous conservons comme notion de taille le nombre de feuilles, nous prouvons le résultat suivant :

Théorème 3.2.1

Dans les systèmes composés soit de l'`implication` *et des k littéraux positifs $\{x_1, \dots x_k\}$, soit des connecteurs* `et/ou` *et des $2k$ littéraux positifs et négatifs, étant donné une fonction f, de complexité $L(f)$, sa probabilité vaut*

$$\mu_k(f) \underset{k\to\infty}{\sim} \frac{\lambda(f)}{k^{L(f)+1}},$$

où $\lambda(f)$ est une valeur ne dépendant que de f (et pas de k).

Notons que si la fonction f est la fonction `vrai` ou `faux`, alors, par définition, nous prenons sa complexité égale à 0. Suivant le système logique, seules les constantes multiplicatives des probabilités des fonctions sont modifiées. De la preuve de ces résultats, nous en déduisons le comportement informel suivant.

Etant donné une fonction f fixée, un grand arbre la représentant est presque sûrement composé d'une tête, de taille $L(f)$, qui la calcule et dans laquelle est greffé un grand sous-arbre qui ne modifie pas la fonction calculée par la tête de l'arbre.

3.3 Associativité et nouvelle notion de taille

Rappelons que pour les derniers modèles, munis de la propriété d'associativité, nous avons conservé la notion de taille classique du contexte des expressions booléennes, c'est-à-dire le nombre d'occurrences des littéraux, ou de façon équivalente le nombre de feuilles dans les expressions arborescentes. Cette notion de taille est complètement naturelle dans le contexte d'arbres binaires puisque le nombre de nœuds dépend uniquement du nombre de feuilles.

Nous considérons, dans cette section associée à l'article [9], le système de logique `et/ou` avec les $2k$ littéraux $\{x_1, \bar{x}_1, \dots, x_k, \bar{x}_k\}$ et muni de la règle d'associativité. Dans une telle expression les connecteurs sont stratifiés. Les connecteurs sont choisis non commutatifs. Notons néanmoins, que les résultats et preuves pourraient être adaptés au cas commutatif.

Étant donné une expression booléenne (arborescente), la *taille* de l'expression correspond au *nombre total de nœuds* dans l'arbre. L'arbre à droite de la Figure 3.4 est une expression dans ce modèle. Sa taille vaut (dans ce nouveau contexte) 10, et il calcule la fonction $(x_1, \dots, x_k) \mapsto x_1 \vee x_2 \vee (\bar{x}_3 \wedge x_4 \wedge x_8)$. La complexité de cette fonction vaut 7 (les connecteurs identiques sont fusionnés dans l'expression arborescente), et les arbres minimaux ne contiennent que 5 feuilles.

On définit toujours la complexité des fonctions constantes `vrai, faux` comme égale à 0.

Théorème 3.3.1

Dans ce système `et/ou` *avec $2k$ littéraux, et la nouvelle notion de taille, étant donné une fonction f de complexité $L(f) \geqslant 3$, sa probabilité vaut*

$$\mu_k(f) \underset{k\to\infty}{\sim} \frac{\lambda(f)}{k^{L(f)}},$$

où $\lambda(f)$ est une valeur ne dépendant que de f (et pas

de k). Par ailleurs, la probabilité des constantes `vrai` *et* `faux` *est une constante strictement positive, et si f est une projection (i.e. il existe i tel que $f(x_1, \ldots, x_k) = x_i$), alors $\mu_k(f) \sim \lambda(f)/k^2$.*

Notons qu'il n'y pas de fonction de complexité valant 2.

Bien que ce résultat semble très similaire aux distribution de probabilité décrites par les précédents modèles pour lesquels la notion de taille était distincte, cette dernière distribution est encore davantage biaisée vers les deux fonctions constantes `vrai` et `faux`. Par rapport au modèle associatif dont la taille associée est les nombre de feuilles, les expressions sont les mêmes, donc dans ce nouveau modèle les complexités des fonctions sont augmentées (puisque l'on compte également les connecteurs dans la taille), donc l'ordre de grandeur des probabilités des fonctions autres que les constantes (et les projections) sont multipliées par un facteur inférieur $1/k$. La probabilité des constantes compense la diminution des probabilités des autres fonctions.

En dehors de cette remarque purement calculatoire, il est à noter que les expressions typiques de ce nouveau modèle ont une forme bien différente de celle des autres modèles, asymptotiquement pour n puis k tendant vers l'infini. En effet, une des différences les plus importantes est relative au nombre de feuilles attachées directement à la racine d'un arbre de grande taille. Alors que ce nombre était contant pour les premiers modèles, dans ce nouveau modèle il y a $2\sqrt{2n}$ feuilles en moyenne au premier niveau de l'arbre. Ce résultat se comprend intuitivement bien. Pour le choix d'étiquetage des feuilles, nous avons $2k$ possibilités, alors que pour les nœuds internes, nous ne disposons que de deux connecteurs. Ainsi, les structures typiques d'arbres comporteront le plus petit nombre de nœuds internes possible. Pour avoir une grande expression, il faut au moins un connecteur à la racine ; si nous mettons beaucoup de feuilles au niveau 1, il y a de forte chances d'obtenir une expression calculant soit `vrai`, soit `faux` suivant le connecteur étiquetant la racine.

Une seconde particularité de ce modèle concerne les fonctions de grande complexité. Soit f_0 une fonction fixée (i.e. sa complexité est indépendante de k). Définissons $\mathfrak{F}_{\geqslant f_0}$ l'ensemble des fonctions g (sur k variables) qui s'évaluent à `vrai` pour toutes les évaluations pour lesquelles f_0 s'évalue à `vrai`. En particulier, informellement, des fonctions dont la complexité dépend de la plupart des variables sont contenues dans $\mathfrak{F}_{\geqslant f_0}$. Mais la fonction `vrai` appartient également à ce dernier ensemble et nous prouvons le résultat suivant

$$\mu_k\left(\mathfrak{F}_{\geqslant f_0}\right) \underset{k\to\infty}{\sim} \mu_k\left(\texttt{vrai}\right).$$

En particulier, la probabilité des fonctions de grande complexité de $\mathfrak{F}_{\geqslant f_0}$ tend vers 0, lorsque n et k tendent successivement vers l'infini. Ce résultat n'est pas prouvé dans les modèles précédents.

3.4 Modèle arborescent équilibré

Ainsi, nous avons remarqué qu'en utilisant des structures arborescentes variées (plane, non-plane, binaire, non-binaire), en prenant différents systèmes logiques, les résultats sont robustes dans le sens où ils sont très similaires (même si les preuves sont différentes d'un point de vue technique). De façon macroscopique, tous ces modèles se comportent de la même manière.

En étudiant un article de Valiant [Val84], nous en déduisons que nous pourrions peut-être perturber ces modèles très similaires en modifiant la structure arborescente de façon drastique.

En considérant spécifiquement la forme des arbres, une des plus régulières qui soit est l'arbre équilibré, que nous définissons par l'arbre binaire ayant toutes ses feuilles au même niveau, i.e. à la même profondeur. Nous notons immédiatement que, pour une taille donnée, le nombre de tels arbres est très restreint par rapport à l'ensemble des arbres. En effet, pour une taille n donnée, il y a exactement une structure arborescente équilibrée (sans étiquette), lorsque n est une puissance de 2, contre C_{n-1} arbres binaires au total (et aucun arbre équilibré si n n'est pas une puissance de 2). Ainsi nous nous attendons à obtenir des résultats bien différents pour l'étude de la fraction limite des arbres équilibrés par rapport à celle des arbres sans forme particulière.

La classe d'arbres obtenue en ne conservant que les arbres équilibrés a été étudiée en détails sous deux noms distincts : *amplification probabiliste* et *processus de croissance*. Valiant a initié ce champ de recherche en 1984 [Val84]. Son but a été de construire une expression de petite taille calculant la fonction *majorité* sur k variables Booléennes. Celle-ci vaut `vrai` pour une affectation a, si et seulement si au moins la moitié des variables de a valent `vrai`. Évidemment, le problème est intéressant lorsque le nombre de variables k devient grand. Au fil des années la méthode de Valiant a été élargie afin d'obtenir le même genre de résultats pour d'autres fonctions. En particulier, Boppana [Bop85] généralise la méthode de Valiant. Il obtient des petites expressions pour chacune des *fonctions seuil* : une généralisation de *majorité*.

Nous nous sommes intéressés, dans l'article [16] aux arbres `et/ou` équilibrés. Afin de construire nos arbres, dans un premier temps, nous étiquetons indépendamment chaque feuille avec l'un des k littéraux positifs. Dans un second temps, nous montrons que le modèle autorisant les littéraux négatifs est une extension directe du premier modèle. De plus, les littéraux sont choisis d'après une distribution de probabilité μ_0 (non forcément uniforme). Nous choisissons les connecteurs `et` ou `ou` avec une distribution de Bernoulli de paramètre p (pour `et`). Enfin, étant donné $\alpha = (\alpha_1, \ldots, \alpha_k) \in \mathbb{R}^k$ et $\theta \in \mathbb{R}$, la *fonction seuil linéaire* $T_{\alpha,\theta}$ est la fonction Booléenne sur $\{0,1\}^k$ définie par :

$$T_{\alpha,\theta}(a) = 1 \;\Leftrightarrow\; \alpha_1.a_1 + \ldots + \alpha_k.a_k \geqslant \theta.$$

Théorème 3.4.1

Soient $k \geqslant 1$, $p \in [0,1]$, et μ_0 une distribution de proba-

bilité sur $\{x_1, \ldots x_k\}$ telle que pour tout i, $\mu_0(x_i) > 0$. La suite de distributions (π_n) admet la distribution limite suivante :

- *Supposons $p > 1/2$. Le support de la distribution limite est restreint à la fonction $x_1 \wedge \ldots \wedge x_k$. De façon duale, si nous avons $p < 1/2$, alors le support de la distribution limite est restreint à la fonction $x_1 \vee \ldots \vee x_k$.*
- *Si $\wedge$ et $\vee$ ont la même probabilité ($p = 1/2$), alors la distribution limite de probabilité est concentrée sur l'ensemble des fonctions seuil linéaire de la forme $\{T_{\mu_0,\theta} \mid \theta \in \mathbb{R}\}$ de la manière suivante : Soient $\theta_0 = 0 < \theta_1 < \ldots < \theta_s = 1$ les différents poids de tous les points de $\{0,1\}^k$, pour $i \in \{1, \ldots, s\}$, $\pi(T_{\mu_0,\theta_i}) = \theta_i - \theta_{i-1}$.*

Nous nous rendons compte rapidement pour ces modèles qu'une approche davantage *probabiliste* est plus efficace que les approches plus *combinatoires* que nous avons utilisées jusqu'à maintenant.

Corollaire 3.4.2

Pour $p = 1/2$, et μ_0 la distribution uniforme sur $H_0 = \{x_1, \bar{x}_1, \ldots, x_k, \bar{x}_k\}$, la suite π_n a une distribution limite qui est uniforme sur les deux fonctions constantes `vrai` *et* `faux`*.*

Ce premier pas dans le contexte des arbres équilibrés présente une distribution sur les fonctions booléennes bien différente des distributions engendrées par l'ensemble des structures arborescentes : tout d'abord le support est restreint à un petit nombre de fonctions et la distribution est uniforme dès que μ_0 est uniforme.

Chapitre 4

Effet Shannon relatifs aux expressions arborescentes

Dans le système `et/ou`, dont la taille d'une expression correspond à son nombre total de nœuds, nous avons été en mesure de donner des informations sur la probabilité des fonctions dont les expressions minimales dépendent d'un grand nombre de variables. Nous n'avons présenté aucun tel résultat dans les systèmes dont la taille est le nombre de feuilles des expressions.

Peut-on quantifier la probabilité des fonctions de grande complexité dans ces systèmes ?

Une autre direction relative aux fonctions de grande complexité consiste à déterminer quel est, asymptotiquement, le support essentiel de la distribution de probabilité. Les résultats que nous avons présentés, concernant les probabilités des fonctions fixées (dont le nombre de variables est fixe, indépendant de k) permettent d'affirmer, par exemple, que l'ensemble de ces fonctions a une probabilité qui tend vers 0, lorsque k tend vers l'infini.

Est-on en mesure d'extraire une sous-famille de fonctions, non triviale, qui contienne le support de la distribution limite sur les fonctions booléennes ?

4.1 Effet Shannon

Avant de se plonger dans cette question, rappelons des résultats concernant la distribution uniforme sur l'ensemble des fonctions booléennes (à k variables). Les premiers travaux dans ce contexte remontent aux années 1940, en particulier aux articles [RS42, Sha49] de Riordan et Shannon.

En considérant la distribution uniforme sur les fonctions booléennes à k variables, Lupanov [Lup62, Lup65] a prouvé que, asymptotiquement, les fonctions, représentées en des arbres binaires, ont une complexité presque sûrement de l'ordre $2^k/\log_2 k$. Originellement, Shannon s'est intéressé aux fonctions représentées via des circuits, et a prouvé le premier résultat dans ce sens. Par la suite, nous utiliserons la notion suivante.

Définition 4.1.1

Une suite de distributions $(\nu_k)_{k\geqslant 1}$, définies respectivement sur les ensembles de fonctions booléennes à k variables, exhibe l'effet Shannon *si, pour tout $\varepsilon > 0$,*

$$\lim_{k\to\infty} \nu_k\left(\left\{f \text{ such that } L(f) \geqslant \frac{2^k(1-\varepsilon)}{\log_2 k}\right\}\right) = 1.$$

Une preuve synthétique du résultat de Lupanov, basée sur des arguments combinatoires, est présentée dans le livre de Flajolet et Sedgewick [FS09, p. 77]. L'argument central consiste à observer que le nombre d'expressions constructibles avec les $2k$ littéraux de tailles inférieures à $2^k/\log_2 k$ est négligeable devant le nombre de fonctions à k variables, lorsque k tend vers l'infini. Par conséquent essentiellement toutes les fonctions ont une complexité supérieure à $2^k/\log_2 k$.

Notons que Lupanov [Lup62] a par ailleurs prouvé que la complexité maximale d'une fonction booléenne sur k variables est inférieure ou égale à $\frac{2^k}{\log_2 k}(1 + o(1))$. Par conséquent, lorsque k tend vers l'infini, la suite des distributions uniformes $(\nu_n)_{n\geqslant 1}$ exhibe l'effet Shannon, mais de plus, presque sûrement toutes les fonctions sont de complexité presque maximale (i.e. maximale en première approximation).

Afin de prouver qu'une suite de distributions $(\nu_n)_{n\geqslant 1}$ n'exhibe pas l'effet Shannon, il suffit de trouver une fonction sous-exponentielle (en k), que nous noterons $g(k)$ et une constante $\alpha > 0$ telles que, il existe un rang N, tel que pour tout $n \geqslant N$,

$$\nu_n\left(\{f \text{ telle que } L(f) \leqslant g(n)\}\right) \geqslant \alpha.$$

Une telle approche est bien différente de celles présentées dans la Section 3.1 puisque l'on doit ici considérer un ensemble de fonctions booléennes dont les complexités dépendent de k. Toutefois, dans certains cas, le résultat est direct. En effet, en considérant le système `et/ou` avec $2k$ littéraux, et la notion de taille correspondant au nombre total de nœuds des expressions, les fonctions constantes `vrai` et `faux` ont toutes les deux, asymptotiquement, une probabilité strictement positive, donc ce modèle particulier n'exhibe pas l'effet Shannon.

4.2 Distributions sur les fonctions

Pour la suite de ce chapitre, nous nous concentrons donc à nouveau sur les différents systèmes logiques (évoqués précédemment) tels que la taille d'une expression est égale au nombre d'occurrences de variables (avec ou non les règles d'associativité ou commutativité). Les résultats à venir sont extraits de nos deux articles [17] et [20].

Proposition 4.2.1

Si K est une constante indépendante de k, alors, aucun ensemble de fonctions booléennes de complexités au plus K n'a une probabilité suffisante pour réfuter l'effet Shannon.

Cette proposition est en fait un corollaire des Théorèmes 3.1.1 et 3.2.1.

Théorème 4.2.2

Les distributions de probabilité induites par les expressions arborescentes dans les systèmes logiques de l'`implication` *et* `et/ou` *n'exhibent pas l'effet Shannon.*

Bien que les preuves dans chacun des deux modèles soient basées sur des raisonnements similaires, la partie technique est nettement plus délicate dans le contexte des arbres `et/ou` que dans celui de l'`implication`. Par conséquent, alors que dans ce dernier cas nous avons pu définir une famille de fonctions et calculer explicitement une borne inférieure de sa probabilité, dans le cas des arbres `et/ou`, nous nous sommes contentés de réfuter l'effet Shannon, sans tenter d'obtenir une borne fine de la probabilité de la famille de fonctions considérées : nous avons simplement prouvé que cette probabilité est strictement positive.

Démonstration (idées) : Fixons L une valeur dépendant du nombre k de variables. En considérant l'ensemble $\mathcal{E}$ des arbres de taille au plus L, nous savons que les fonctions calculées par ces expressions ne peuvent pas être de complexité supérieure à L. Dès lors, nous utilisons les notions d'expansions introduites dans la Section 3.1. Notons que dans le cas des arbres `et/ou`, bien que les expansions ne soient pas introduites, que ce soit dans l'article de Kozik [Koz08] ou dans notre article [5], la notion de langage de motifs qui y est utilisée est une autre manière de les présenter : ce sont les mêmes idées qui guident les deux approches (cf. l'article [20]). Afin de construire une famille de grands arbres (i.e. de taille arbitrairement grande), nous faisons des expansions dans les arbres de $\mathcal{E}$. Ainsi les fonctions calculées sont toujours les mêmes que celles associées à $\mathcal{E}$. Dans un arbre de taille $\ell \geqslant L$, nous pouvons effectuer $2\ell - 1$ expansions de type tautologie (une en chaque nœud de l'arbre). Effectuer deux expansions dans le même arbre n'apporte essentiellement pas de nouvelles expressions. Par contre, le nombre d'expansions de type but ou prémisse est un peu plus complexe à calculer. En effet, en certain nœuds de l'arbre il est possible de réaliser plusieurs expansions de type but avec des variables distinctes. Par exemple dans l'arbre de gauche de la Figure 3.1, il est possible d'effectuer, dans le père de la feuille étiquetée par x_2, deux expansions de type but, l'une avec la variable x_1 et l'autre avec la variable x_3.

Ainsi, nous prouvons que lorsque la taille ℓ des arbres tend vers l'infini, en moyenne dans un arbre de taille ℓ, nous pouvons effectuer $\Theta(\ell^{3/2})$ expansions de type prémisse ou but. En particulier, le nombre d'expansions de type tautologie devient négligeable.
Toutefois, la preuve n'est pas entièrement terminée. Rappelons que nous considérons tous les arbres de taille inférieure à L, même ceux qui ne sont pas des arbres minimaux : par exemple, ils pourraient déjà contenir une expansion. Ainsi, si nous ne voulons pas compter plusieurs fois le même arbre (après avoir effectué notre expansion), il faut être en mesure de distinguer ce sous-arbre greffé. Par conséquent, nous allons restreindre la structure des expansions que nous greffons dans l'arbre originel. Une manière efficace consiste à construire des expansions dont les deux enfants de la racine sont suffisamment grands (chacun de taille supérieur à L). En choisissant correctement la valeur L, d'ordre $\Theta(k^2)$ pour l'`implication` et $\Theta(k^{2+\epsilon})$ pour le modèle `et/ou`, nous prouvons que l'ensemble de fonctions de $\mathcal{E}$, de complexités inférieures à L, a une probabilité strictement positive. Donc il n'y a pas d'effet Shannon. ■

Ce théorème affirme qu'il existe une classe de fonctions de petite complexité (quasi quadratique en le nombre de variables) dont la fraction limite est une constante strictement positive. Mais nous ne sommes pas en mesure actuellement de caractériser une famille (non-triviale) de fonctions dont la probabilité vaut 1. En particulier, *en choisissant aléatoirement une fonction booléenne avec sa probabilité induite par la distribution des structures arborescentes, cette fonction a-t-elle presque sûrement une complexité polynomiale en le nombre de variables ?*

Chapitre 5

Contexte de la satisfaisabilité

Le but de ce chapitre est de répondre à la question suivante. *Prenons aléatoirement une expression booléenne de taille n, construite dans un système logique dépendant de k_n variables. Quelle est la probabilité que cette expression soit satisfaisable, c'est-à-dire qu'il existe une affectation de ses variables qui la rende vraie, lorsque la taille n tend vers l'infini ?*

Dans les modèles présentés dans les Chapitres 3 et 4, nous nous sommes restreints à une suite $(k_n)_{n\geqslant 1}$ constante égale à k. Ce chapitre présente une rupture fondamentale avec les précédents modèles, afin de généraliser les résultats à des suites $(k_n)_{n\geqslant 1}$ raisonnables.

Pour les modèles pour lesquels le nombre de variables est fixé à k, il ne semble pas clair au premier abord que l'on puisse laisser croître la valeur de k lorsque n grandit. Dans les modèles précédents, nous faisions tendre la taille n vers l'infini (à k fixé), puis dans un second temps nous laissions k tendre vers l'infini. Bien entendu, l'interversion des limites n'est pas justifiable mathématiquement.

Rappelons toutefois qu'un premier modèle, avec un nombre infini de variables, a été présenté dans le contexte des tautologies (cf. Chapitre 2). Afin de mener l'étude quantitative, nous nous sommes intéressés aux expressions équivalentes à renommage des variables près. L'article [21], étendu avec la version [7], consiste à adapter le système classique `et/ou` pour cette relation d'équivalence.

5.1 Problème général de la satisfaisabilité

Depuis plusieurs décennies, des informaticiens, des mathématiciens mais aussi des physiciens s'intéressent au contexte de la satisfaisabilité des fonctions booléennes aléatoires. L'un des problèmes le plus étudié est dénommé le modèle 3–SAT. Il consiste à choisir uniformément une expression construite sur la conjonction de n clauses, chacune étant une disjonction de 3 littéraux positifs ou négatifs parmi k_n variables. Voici une expression 3–SAT :

$$(x_1 \vee \bar{x}_2 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_1 \vee \bar{x}_4)\,.$$

Cette expression (de taille 3) est satisfaisable car il suffit d'affecter x_1, x_2 et x_4 à `faux` afin que l'expression s'évalue à `vrai`. Par contre, il ne s'agit pas d'une tautologie : en affectant x_1 et x_4 à `faux` et x_2 à `vrai`, l'expression s'évalue à `faux`.

Une des raisons de l'intérêt autour de ce problème résulte du fait qu'il est NP-complet, tel le problème général SAT (cf. par exemple [AB09]).

La question de déterminer si une grande expression est satisfaisable est partiellement résolue. L'article [AM06] prouve, par exemple, qu'il existe une transition de phase suivant la valeur de la suite (k_n) : informellement, lorsque le rapport k_n/n est suffisamment petit, alors l'expression est satisfaisable avec probabilité tendant vers 1 alors que si le rapport est trop grand, cette probabilité tend vers 0. La transition de phase fait l'étude de nombreux articles à elle seule.

Par ailleurs, de nombreux autres problèmes autour de la satisfaisabilité ont été abordés. Nous n'en citerons que quelques-uns, choisis de par leur proximité à la combinatoire analytique en particulier. Le problème 2-XORSAT a été étudié par Daudé et Ravelomanana [DR11], en particulier dans le but d'exhiber et de décrire précisément sa transition de phase. Il a aussi été revisité [dPGGK16] afin d'en obtenir une étude fine de la distribution de probabilité induite sur les fonctions booléennes.

5.2 Arbres booléens pour la satisfaisabilité

Le but de cette section consiste à étendre les modèles arborescents booléens, présentés dans les chapitres précédents, afin de pouvoir s'intéresser à des questions naturelles dans le contexte de la satisfaisabilité tout en continuant à utiliser des outils de combinatoire analytique, et éventuellement les premières approches que nous avons mises en place pour la logique quantitative.

Dans l'article [7], nous nous sommes concentrés sur le système logique `et/ou` (avec pour taille le nombre d'occurrences de littéraux dans les expressions). Toutefois, nous sommes convaincus que l'approche présentée s'adapterait naturellement au système de l'`implication` et donnerait des résultats similaires.

Comme nous l'avons vu précédemment dans les modèles originels d'expressions arborescentes, afin d'obtenir les résultats quantitatifs, nous avons procédé à deux passages successifs à la limite infinie. Dans un premier temps,

la fraction limite correspond à la limite des ratios des cardinalités des d'ensembles d'expressions dont la taille tend vers l'infini. Dans un second temps, pour obtenir une information quantitative, on laisse croître le nombre de variables k vers l'infini.

Nous présentons ici deux modèles permettant d'éviter la double limite, tout en obtenant des résultats quantitatifs (exploitables). Nous nous basons sur une suite $(k_n)_{n\geqslant 1}$ croissante d'entiers telle que k_n tende vers l'infini lorsque n tend vers l'infini.

Le premier modèle, noté $(\mathcal{G})$, est une généralisation du système logique `et/ou` construit avec k_n variables (plus exactement $2k_n$ littéraux). Nous définissions le modèle $(\mathcal{G})$ ainsi :

(i) considérons la distribution uniforme sur l'ensemble des arbres binaires `et/ou` ayant n feuilles étiquetées via l'ensemble $\{x_1, \bar{x}_1, \ldots, x_{k_n}, \bar{x}_{k_n}\}$;

(ii) notons $\mathbb{P}_n$ la distribution induite sur l'ensemble des fonctions booléennes.

Nous remarquons que la taille A_n de l'ensemble des arbres `et/ou` à n feuilles vaut

$$A_n^{(\mathcal{G})} = 2^{n-1}(2k_n)^n \cdot \mathrm{Cat}_{n-1},$$

avec Cat_n le n-ième nombre de Catalan. Remarquons que pour chaque n fixé, toutes les quantités sont finies, donc il n'y a pas de problème de définition de classe combinatoire. On définit pour chaque fonction booléenne f sa suite de probabilités via le rapport :

$$\mathbb{P}_n(f) = \frac{A_n^{(\mathcal{G})}(f)}{A_n^{(\mathcal{G})}}.$$

Dans ce contexte, nous conservons la notion classique de la taille d'une expression (son nombre de feuilles) et aussi de la complexité $L(f)$ d'une fonction f. On prouve le théorème suivant.

Théorème 5.2.1 (Modèle $(\mathcal{G})$)

Soit $(k_n)_{n\geqslant 1}$ une suite croissante d'entiers tendant vers l'infini lorsque n tend vers l'infini. Pour tout fonction booléenne f, il existe une constante positive $\alpha_f^{\mathcal{G}}$ telle que, asymptotiquement lorsque n tend vers l'infini,

$$\mathbb{P}_n(f) \sim \alpha_f^{(\mathcal{G})} \cdot \left(\frac{1}{k_n}\right)^{L(f)+1}.$$

À partir de ce résultat nous obtenons le corollaire suivant concernant le problème Catalan-SAT (ce nom, que nous avons choisi, résulte de la forme arborescente des expressions considérées).

Corollaire 5.2.2 (Catalan-SAT)

Soit $(k_n)_{n\geqslant 1}$ une suite croissante d'entiers tendant vers l'infini lorsque n tend vers l'infini. Soit une expression, de taille n, choisie uniformément dont les feuilles sont étiquetées avec $\{x_1, \bar{x}_1, \ldots, x_{k_n}, \bar{x}_{k_n}\}$. Cette expression est satisfaisable avec probabilité tendant vers 1 lorsque n tend vers l'infini.

Le second modèle $(\mathcal{E})$ permettant d'éviter la double limite consiste à considérer les expressions à renommage (cohérent) près des variables. Nous avons mentionné un tel modèle dans le contexte des tautologies (Chapitre 2). Nous allons le formaliser davantage ici, afin de pouvoir étudier la probabilité des classes d'équivalence de fonctions. L'idée fondamentale est la suivante : les deux fonctions $(x_i)_{i\geqslant 1} \mapsto x_1 \wedge x_2$ et $(x_i)_{i\geqslant 1} \mapsto x_{38} \wedge \bar{x}_{12}$ sont vues comme deux réalisations de la même fonction `conjonction binaire`. Informellement, deux arbres `et/ou` sont équivalents si les feuilles du premier arbre peuvent être réétiquetées sans collision afin d'obtenir le second arbre. Voila la définition plus formelle : Soit A et B deux arbres `et/ou`. Les arbres A et B sont équivalents si

(i) leur structure arborescente est la même ;

(ii) deux feuilles sont étiquetées par la même variable (littéral positif ou négatif) dans A si et seulement si ces feuilles-là sont étiquetées par la même variable dans B ;

(iii) deux feuilles sont étiquetées par le même littéral dans A si et seulement si ces feuilles-là sont étiquetées par le même littéral dans B.

Cette relation d'équivalence sur les expressions induit naturellement une relation d'équivalence sur les fonctions booléennes. Une variable x est dite *essentielle* pour une fonction f, si les restrictions de la fonction lorsque x est `vrai` et `faux` sont deux fonctions distinctes. Par exemple, dans l'arbre à gauche de la Figure 3.2, les variables x_1, x_2, x_3 et x_4 sont essentielles, mais x_7 ne l'est pas. Remarquons que deux fonctions équivalentes dépendent du même nombre de variables essentielles et ont la même complexité.

Notons $\langle f \rangle$ la classe d'équivalence de la fonction f, $L\langle f\rangle = L(f)$ est la complexité des fonctions de $\langle f\rangle$ et $E\langle f\rangle = E(f)$ est le nombre de variables essentielles de $\langle f\rangle$. Pour une classe $\langle f\rangle$, nous définissons sa *multiplicité* comme l'entier $R\langle f\rangle = L\langle f\rangle - E\langle f\rangle$. Il correspond au nombre de répétitions des variables dans les arbres minimaux de la classe $\langle f\rangle$.

Rappelons que la suite $(k_n)_{n\geqslant 1}$ est croissante et tend vers l'infini avec n. Dans la suite nous ne considérons que les classes d'équivalence contenant au moins un élément dont les feuilles sont étiquetées uniquement avec l'ensemble $\{x_1, \bar{x}_1, \ldots, x_{k_n}, \bar{x}_{k_n}\}$.

De façon analogue au premier modèle nous avons

$$A_n^{(\mathcal{E})} = \mathrm{Cat}_n \cdot \sum_{p=1}^{k_n} \left\{ {n \atop p} \right\} 2^{2n-1-p}, \quad \text{et} \quad \mathbb{P}_n\langle f\rangle = \frac{A_n^{(\mathcal{E})}(f)}{A_n^{(\mathcal{E})}},$$

avec $\left\{ {n \atop p} \right\}$ le nombre de Stirling de seconde espèce (cf. par exemple [FS09, p. 735]).

Pour ce modèle, nous prouvons le résultat suivant :

Théorème 5.2.3 (Modèle ($\mathcal{E}$))

Soit $(k_n)_{n\geqslant 1}$ une suite croissante d'entiers tendant vers l'infini lorsque n tend vers l'infini. Il existe une suite $(M_n)_{n\geqslant 1}$ telle que $M_n \sim_{n\to\infty} \frac{n}{\ln n}$, lorsque n tend vers l'infini. De plus, pour toute classe $\langle f\rangle$, il existe une constante positive $\alpha^{\mathcal{E}}_{\langle f\rangle}$ vérifiant :

(i) si pour tout n suffisamment grand, $k_n \leqslant M_n$ alors, asymptotiquement, lorsque n tend vers l'infini,

$$\mathbb{P}_n\langle f\rangle \sim \alpha^{(\mathcal{E})}_{\langle f\rangle} \cdot \left(\frac{1}{k_{n+1}}\right)^{R\langle f\rangle+1};$$

(ii) si, pour tout n suffisamment grand, $k_n \geqslant M_n$, alors, asymptotiquement, lorsque n tend vers l'infini,

$$\mathbb{P}_n\langle f\rangle \sim \alpha^{(\mathcal{E})}_{\langle f\rangle} \cdot \left(\frac{\ln n}{n}\right)^{R\langle f\rangle+1}.$$

Concernant le problème de la satisfaisabilité, nous obtenons le même corollaire que précédemment (cf. Corollaire 5.2.2).

Ce phénomène de seuil est très intéressant et peut facilement se formuler (informellement) : *en ajoutant de plus en plus de variables dans un système, on biaise la distribution sur les classes de fonctions booléennes, jusqu'à un certain point à partir duquel, le fait d'ajouter des variables n'a plus d'impact global sur les probabilités.*

Deuxième partie

Concurrence quantitative : Structures croissantes pour modéliser la sémantique de la concurrence

Chapitre 6

Introduction et préliminaires

Dès les années 1960, les modèles fondamentaux de la théorie de la concurrence font leur apparition. Mentionnons en particulier les réseaux de Petri [Pet62] et les modèles `CSP`, pour *Communicating Sequential Processes*, introduit par Hoare et largement détaillé dans son livre [Hoa85], suivi de `CCS`, *Calculus of Communicating Systems*, présenté par Milner [Mil80]. Ces deux derniers modèles correspondent chacun à une algèbre de processus définissant les opérateurs et concepts fondamentaux pour représenter des systèmes en interactions. Ils visent à étudier le comportement de processus concurrents. Très rapidement une propriété fondamentale de ces systèmes s'impose : *le nombre de comportements d'un processus concurrent fait intervenir une explosion combinatoire.*

Peu d'études se sont intéressées à l'analyse fine de l'explosion combinatoire. Néanmoins, les pires cas ont été analysés, notamment dans le contexte du *model checking*, cf. par exemple [BK08]) pour lequel une analyse exhaustive de l'espace d'états est nécessaire. Une autre étude, plus fine et dans le contexte de la combinatoire analytique, a été proposée par Françon [Fra86], en 1986. Il s'est intéressé à mesurer le degré de parallélisme d'un système composé de processus se partageant un nombre limité de ressources. Un modèle similaire, plus algébrique, a été également étudié dans le contexte du monoïde de traces. Le premier article dans cette direction a été présenté par Saheb [Sah89] en 1989.

En dehors de ces deux directions de recherche, les approches quantitatives se sont pour l'essentiel, contentée de cette très vague notion d'explosion combinatoire omniprésente en concurrence. Nous avons choisi de travailler sur le modèle `CCS` de Milner afin de réaliser une étude fine de la combinatoire résultant de la concurrence. Ce modèle étant un modèle opérationnel, son approche via la combinatoire analytique en est naturelle.

Voilà un exemple simple explicitant la croissance explosive. La syntaxe du processus est la suivante : le processus est composé de n actions en parallèle, toutes indépendantes les unes des autres. L'espace d'états, qui correspond à l'aspect sémantique du processus, contient l'ensemble des comportements possibles du processus, c'est-à-dire l'ensemble des ordres d'exécution des actions de taille $n!$. Ce premier constat établi, de nombreuses approches ont été présentées : *comment analyser un système faisant intervenir un tel espace d'états ?*

Du point de vue quantitatif : *le pire cas est-il réaliste, dans le sens suivant : un processus en pratique a-t-il un nombre d'exécutions proche du pire cas ? Si oui, est-il réaliste de tenter d'extraire des informations d'un espace d'états de taille exponentielle, voire factorielle ? Quelle est l'explosion engendrée par un grand processus typique ? Mais, un processus typique, de quoi s'agit-il ?*

Un problème central de la concurrence réside dans le fait de vérifier que certaines propriétés sont satisfaites par chaque exécution. En effet, les processus concurrents ont régulièrement des comportements inattendus et à proscrire, par exemple des interblocages entre plusieurs sous-processus, ou *deadlocks*. Le parcours exhaustif de l'ensemble de l'espace d'états (i.e. de toutes les exécutions) est une technique appelée *model checking* (cf. par exemple [CGP99]). Néanmoins il semble évident que cette approche ne passe pas à l'échelle lorsque la taille des processus augmente. Ainsi, afin de vérifier un système, plusieurs points algorithmiques semblent incontournables.

Du point de vue algorithmique : *étant donné un processus, peut-on dimensionner le nombre de comportements des processus ? Un processus peut-il être analysé, non pas explicitement mais statistiquement ? Peut-on en analyser seulement une partie des comportements, sans introduire de biais dans le choix de ces comportements ?*

C'est pour répondre à ce type de questions quantitatives et algorithmiques, que nous avons commencé par quantifier de façon très fine et avec des méthodes élaborées, l'explosion combinatoire de modèles de plus en plus aboutis issus de la concurrence. Nous estimons cette première étape quasiment incontournable, pour obtenir dans un second temps des approches algorithmiques efficaces sur des processus fixés : par exemple dimensionner son espace d'états, ou effectuer de la génération aléatoire uniforme d'exécutions ou de leurs préfixes. Notre approche se justifie a posteriori en remarquant que nos descriptions efficaces d'algorithmes reposent sur la maîtrise que nous nous sommes faites de la combinatoire associée aux processus de plus en plus expressifs.

À travers nos études, nous avons aussi eu pour but d'enrichir les outils de combinatoire analytique, en particulier les structures (arborescentes, ou de graphes) éti-

quetées avec contraintes. Ainsi, pour pouvoir se concentrer sur certaines structures combinatoires, nous avons accepté en conséquence d'introduire certaines restrictions dans les modèles de concurrence. Néanmoins, l'un de nos buts originels consiste à comprendre quels opérateurs fondamentaux engendrent l'explosion combinatoire classique du modèle `CCS`. Trois concepts semblent incontournables : il s'agit de la mise en parallèle de plusieurs processus, du choix non-déterministe de plusieurs processus et de la synchronisation de plusieurs processus. Les comprendre d'un point de vue combinatoire permet, par exemple, d'envisager des manières de contrer l'essor de l'espace d'états, mais aussi de produire des algorithmes, ad hoc, efficaces en moyenne (suivant des distributions adéquates). Notre approche consiste à étudier les modèles de façon incrémentale, en ajoutant les opérateurs ou concepts les uns après les autres afin de comprendre leurs interactions et d'adapter nos outils permettant le comptage d'exécutions (résultats quantitatifs) et la génération aléatoire d'exécutions (notamment pour réaliser de la vérification statistique).

Dans ce mémoire, nous nous concentrons davantage sur les structures combinatoires, en exhibant leur signification pour la concurrence, plutôt que l'inverse : nous proposons ainsi une justification a posteriori de nos modèles. Nous avons réalisé ce choix pour la raison suivante : *bien que nos études soient centrées sur des problèmes issus de la concurrence, les méthodes proposées font intervenir des structures combinatoires complexes, naturelles, et plutôt mal connues (ou même inconnues) et nous souhaitons en conséquence, ici, les mettre en avant, les étudier combinatoirement et ensuite les appliquer à la concurrence tout en mettant en perspective d'autres domaines dans lesquels elles pourraient être appliquées (par exemple, la théorie des ordres partiels).*

Deux structures sont omniprésentes dans toute cette partie. Étant donné un processus, nous avons sa *représentation syntaxique* qui correspond à la manière dont il est fondé. Cette représentation syntaxique est basée sur une composition de règles permettant de construire des processus dont la taille est de plus en plus grande. Dans nos premiers travaux, l'objet syntaxique est construit à partir d'une spécification, c'est un objet décomposable (dans les termes de Flajolet et Sedgewick [FS09]). Ce n'est plus tout à fait le cas lorsque nous nous intéressons aux processus en toute généralité. La plupart du temps, l'objet syntaxique a une structure d'arbre ou de graphe acyclique dirigé (DAG). La seconde structure fondamentale est la *représentation sémantique* associée à la syntaxe d'un processus. Cette structure sémantique est un encodage de l'ensemble des comportements valides du processus, c'est-à-dire l'ensemble de ses exécutions possibles (ou ordres d'exécution des actions qui le composent). Dans cette partie du mémoire, cette structure sémantique est arborescente : les préfixes communs des exécutions sont partagés. Dans la partie concernant les perspectives de notre travail, Section 13.2, nous introduisons brièvement la notion de DAG sémantiques qui sont obtenus via un second partage : le partage des suffixes communs des exécutions.

Au fil de cette partie dédiée à la concurrence d'un point de vue quantitatif, nous allons étudier les modélisations combinatoires de la syntaxe et de la sémantique induites par le *parallélisme*, dans un premier temps, le *non-déterminisme*, dans un second temps, puis la *synchronisation* dans un troisième temps. La complexité technique de la combinatoire induite augmentant au fur et à mesure que les concepts de concurrence se mélangent, nous ne serons pas en mesure d'étudier, quantitativement et algorithmiquement, un grand processus quelconque, dans le sens où il est construit sans aucune contrainte sur ces trois piliers de la concurrence. Néanmoins, rappelons le résultat de Brightwell et Winkler [BW91] qui, adapté au contexte de la concurrence, affirme que le comptage des exécutions d'un processus concurrent, en toute généralité, est un problème #P-complet. Bien que ce résultat ne donne pas d'information sur les études quantitatives en moyenne, nous avons tout de même choisi des études de sous-classes de processus aussi bien pour les aspects quantitatifs qu'algorithmiques.

Remarquons que dans les études que nous avons menées dans nos articles, et ce, pour l'ensemble des modèles, notre but a été de travailler directement sur la structure syntaxique en se privant de toute construction sémantique. Cette contrainte très forte, nous évite en particulier toute explosion combinatoire : nos structures annexes, techniques, sont de taille de l'ordre au plus en $O(n^2)$ pour un processus de taille syntaxique n.

Néanmoins, à l'instar de nombreux travaux qui travaillent sur les structures sémantiques (de taille exponentielle !), par exemple [God96, Val89] et d'autres cités dans [Sen07], nous envisageons à court terme (cf. Section 13.2) d'étudier quantitativement si des compressions dans la représentation sémantique, par exemple en exploitant des symétries au sein des exécutions, permettraient de réduire de façon drastique la taille de l'espace d'états.

Enfin, à moyen terme, nous envisageons d'assembler nos briques élémentaires afin de pouvoir présenter un outil de vérification statistique, c'est-à-dire une approche permettant d'explorer les comportements d'un processus, non pas de façon exhaustive, mais de façon probabiliste en ayant le contrôle de la distribution de probabilité sous-jacente, et par conséquent en étant capables de calculer des statistiques par exemple concernant la proportion de l'espace d'état vérifiée : il s'agit de la notion classique de couverture de vérification. Ce but est également visé dans l'article de Denise et al. [DGG$^+$12] mais leur approche se base sur une représentation de la sémantique du système, structure que nous souhaitons éviter de construire.

Finalement, les résultats présentés dans le contexte de la concurrence peuvent être traduits dans la théorie des ordres partiels. En effet, nous modélisons les processus concurrents via des ensembles d'actions auxquels nous associons un ensemble de règles de précédence. C'est ce lien qui nous permet d'obtenir des informations exploitables pour la théorie de la concurrence.

Chapitre 7

Modèles arborescents

Quel est, en moyenne, le nombre d'exécutions d'un processus de taille n ? On se directement rend compte que le pire cas (l'explosion combinatoire la plus grande) correspond à la mise en parallèle de n actions et donc, tous les ordres d'exécution sont valides. Le cas, opposé, dans le sens qui induit le moins d'exécutions valides, correspond au processus en fil : il ne contient aucune action en parallèle. Sa correspondance sémantique associée ne contient qu'une seule exécution. Entre ces deux cas extrémaux, *en choisissant uniformément au hasard un processus, de grande taille, peut-on avoir une idée du nombre d'exécutions de ce processus ?*

Une seconde mesure, plus fine, consiste en l'étude du nombre de préfixes d'exécutions de longueur k dans un processus de taille n ($n \geqslant k$). *Peut-on quantifier comment se comporte l'explosion combinatoire lorsque k évolue (entre 1 et n) ?* Cette analyse nous donne des informations de dimensionnement de la représentation sémantique associée à un processus. En particulier, dans la structure sémantique, nous partageons les préfixes d'exécutions identiques. Cette mesure permet donc de quantifier le taux de partage des exécutions au sein de la sémantique.

7.1 Arbres croissants et processus parallèle

Nous considérons ici le fait que des processus puissent être *mis en parallèle*, et exécutés au fur et à mesure dans n'importe quel ordre tant que les contraintes de précédence internes à chaque processus sont respectées. Par la suite, l'*opérateur de mise en parallèle* sera noté $\|$. Si un processus est composé de deux suites d'actions en parallèle (deux séquences d'atomes d'un point de vue combinatoire), on se rend immédiatement compte qu'une exécution n'est rien d'autre que l'entrelacement des actions de ces deux suites. Nous intégrons naturellement un aspect récursif dans la composition des processus, et ainsi nous nous intéressons en conséquence à l'entrelacement au sein de structures arborescentes. Plus particulièrement, nous voulons dans un premier temps avoir des mesures quantitatives typiques concernant les exécutions de processus construits avec le parallélisme et la séquence. Naturellement, nous allons modéliser ces deux opérateurs, au sein de structures syntaxiques arborescentes.

À travers les articles [18, 8] et [23], nous nous sommes intéressés aux quatre modèles les plus simples dans ce contexte. D'un point de vue structurel, nous avons considéré les arbres généraux (dont les nœuds internes ont une arité arbitraire) et les arbres unaires-binaires. D'un point de vue spatial, nous avons étudié les configurations planes et non planes. Le point de départ général de cette étude en largeur consiste à comprendre si l'un des modèles semble plus pertinent que les autres du point de vue des résultats quantitatifs.

Du côté de la technique, la modélisation via des nœuds d'arité arbitraire a un intérêt majeur : elle permet d'encoder structurellement les deux opérateurs (séquence et parallèle) et donc ne contient qu'un seul type de nœud avec une information atomique. La structure binaire n'étant pas suffisamment riche, deux types de nœuds sont nécessaires pour représenter la syntaxe des processus concurrents. Cet encodage, incontournable, n'apporte cependant pas davantage d'expressivité. De plus, on va s'apercevoir qu'il est la source de problèmes techniques profonds dans l'étude quantitative. Notons toutefois que les deux modèles ne sont pas complètement équivalents d'un point de vue étude en moyenne. Utiliser une structure non binaire correspond à l'utilisation une règle d'associativité entre les sous-processus en parallèle. Ainsi, alors que les deux processus $(P \parallel Q) \parallel R$ et $P \parallel (Q \parallel R)$ sont a priori considérés comme deux processus distincts, si l'on utilise la règle d'associativité de l'opérateur parallèle, ce dernier est considéré d'arité arbitraire (strictement plus grande que 1) et les deux processus précédents sont encodés en un unique processus $P \parallel Q \parallel R$, appelé forme normale de ces processus.

Le modèle qui apparait comme étant le plus simple d'un point de vue combinatoire est le modèle arborescent plan et dont les nœuds sont d'arité arbitraire : de ces faits, il est nommé *processus général plan*[1]. Donnons sa syntaxe formelle puis des raisons qui le rendent plus simples que d'autres modèles.

1. Un processus général n'est pas un processus quelconque. Sa structure est bien plus contrainte.

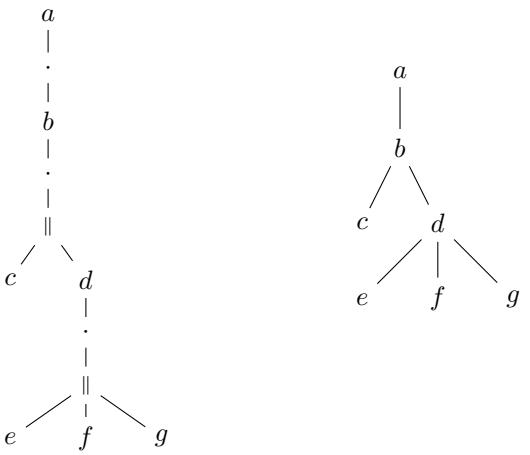


FIGURE 7.1 – *Un processus général (à gauche) et l'arbre syntaxique associé (à droite).*

Définition 7.1.1 (Processus général plan)

- *une* action atomique, *notée* α, *est un processus, appelé* processus atomique,
- *un processus* P, *préfixé par une action* α, *noté* $\alpha.P$, *est un processus, appelé* processus préfixé,
- *la* composition en parallèle $P_1 \parallel \ldots \parallel P_n$ *d'un nombre fini de processus préfixés est un processus.*

La *taille d'un processus* est le nombre d'actions qu'il contient. Chaque action est identifiée à une étiquette unique (une lettre alphabétique). Les étiquettes en elles-mêmes n'ont pas de signification. Autrement dit, deux processus sont équivalents si les étiquettes du premier peuvent être renommées de façon bijective vers celles du second. Notons également que la composition en parallèle est vue comme une suite, c'est-à-dire que l'ordre des sous-processus $P_1, P_2, \ldots, P_n$ est pris en compte.

Commençons par remarquer que cette grammaire définit l'opérateur parallèle comme étant un opérateur associatif (arité arbitraire) et que seules les formes normales sont considérées : deux opérateurs parallèle ne peuvent pas se succéder, dit autrement, dans la composition $P_1 \parallel \ldots \parallel P_n$, aucun des sous-processus P_i n'est une composition en parallèle. De plus, pour les résultats quantitatifs à venir, nous nous restreignons aux processus préfixés de taille n et ne considérons pas les compositions en parallèle de taille n, ces-dernières étant en bijection avec les processus préfixés de taille $n+1$.

Sur la Figure 7.1, on a, à gauche, le processus général

$$a.b.(c \parallel d.(e \parallel f \parallel g)),$$

représenté sous forme arborescente. Du côté droit, on a représenté le modèle combinatoire associé, appelé *arbre syntaxique* : un arbre enraciné plan dont chaque nœud, d'arité arbitraire, est décoré par une action. Le processus, donné en exemple, a pour taille 7. Enfin, notons que ce processus est équivalent au processus $b.a.(r \parallel d.(e \parallel f \parallel g))$, dont seuls les noms d'actions sont distincts.

L'arbre syntaxique contient les actions, et les relations de précédence. Celles-ci sont orientés dans le sens où un nœud μ est dessiné au dessus d'un second nœud ν, et qu'il partagent une arête, alors μ doit être exécuté avant ν. Les arêtes de transitivité ne sont pas représentées. Enfin, notons que pour ce premier modèle, l'arbre syntaxique est plongé dans le plan, donc l'ordre gauche/droite des enfants d'un nœud est important.

Ce premier modèle d'arbre syntaxique est un modèle combinatoire arborescent très classique. Dans le contexte de la combinatoire analytique, on le spécifie via l'équation récursive $\mathcal{G} = \mathcal{Z} \cdot \textsc{Seq}\, \mathcal{G}$. Cette spécification, non ambigüe, décompose un arbre syntaxique (de la classe $\mathcal{G}$) en affirmant qu'un arbre possède une racine (représentée par l'atome $\mathcal{Z}$, dont la taille vaut 1) qui est suivie (éventuellement) par un nombre arbitraire de descendants, qui sont eux-mêmes des arbres syntaxiques de la classe $\mathcal{G}$. Ce modèle combinatoire facile à analyser, rendra les calculs de mesures en moyenne moins techniques que lorsque l'on se base sur une structure syntaxique plus complexe (cf. par exemple les modèles non arborescents du Chapitre 8).

Introduisons désormais la modélisation sans la règle d'associativité : les processus unaires-binaires plans et présentons les principaux changements par rapport au modèle général.

Définition 7.1.2 (Processus binaire plan)

- *une* action atomique, *notée* α, *est un processus, appelé processus atomique*
- *un processus* P, *préfixé par une action* α, *noté* $\alpha.P$, *est un processus, appelé* processus préfixé,
- *la* composition en parallèle $P_1 \parallel P_2$ *de deux processus (quelconques) est un processus.*

Comme dans le modèle général, chaque action ne peut apparaître qu'une seule fois dans un processus. Dans ce modèle, deux processus mis en parallèle ne sont pas forcément préfixés. Comme auparavant, la *taille d'un processus* est le nombre d'actions qu'il contient.

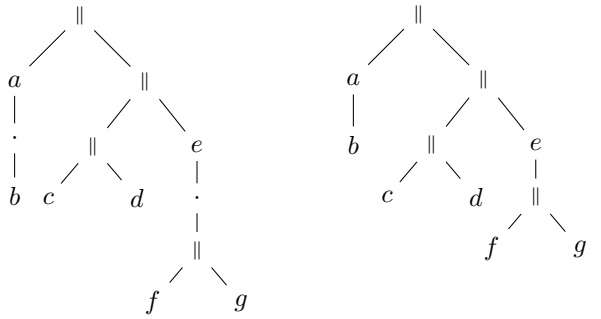


FIGURE 7.2 – *Un processus binaire (à gauche) et l'arbre syntaxique associé (à droite).*

Sur la Figure 7.2, on a, à gauche, le processus binaire,

$$(a.b) \parallel [(c \parallel d) \parallel (e.(f \parallel g))],$$

représenté sous forme arborescente. Du côté droit de la figure, on a représenté l'arbre syntaxique associé. Notons

que dans celui-ci, l'opérateur $\|$ ne peut pas être omis. Une spécification de la classe des arbres syntaxiques binaires est $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} + \mathcal{B} \times \mathcal{B}$, l'atome $\mathcal{Z}$ marquant les actions. Les nœuds contenant l'opérateur $\|$ ne sont pas marqués, ne comptant pas dans la taille.

La mesure fondamentale pour un processus donné, est le nombre de ses exécutions, en définissant une exécution comme étant un ordonnancement de ses actions vérifiant les contraintes de précédence. On a ainsi un premier résultat, à la base des analyses quantitatives des processus concurrents.

Proposition 7.1.3

Soit P un processus (général ou binaire). Le nombre d'exécutions de P est égal au nombre d'étiquetages croissants des actions de son arbre syntaxique.

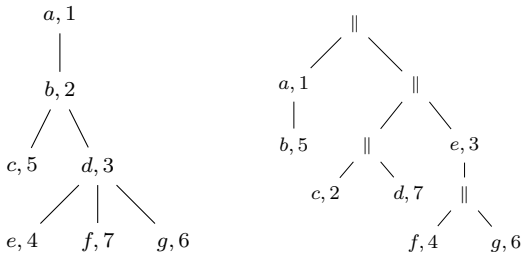


FIGURE 7.3 – *Une exécution des deux processus (général et binaire).*

Notons qu'une structure combinatoire de taille n est dite étiquetée croissante si chacun de ses atomes est associé à un entier entre 1 et n, chaque entier apparaît dans la structure, et les chemins de la racine aux feuilles de la structure contiennent tous une suite d'entiers croissante.

Les deux exécutions des processus représentées sur la Figure 7.3 correspondent aux exécutions $\langle a, b, d, e, c, f, g\rangle$ et $\langle a, c, e, f, b, g, d\rangle$ des processus général et binaire donnés en exemple (cf. Figures 7.1 et 7.2).

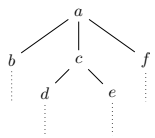
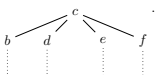
Dans un premier temps, concernant les résultats quantitatifs, nous remarquons que les deux modèles se comportent probablement différemment en moyenne. En effet, il n'y a pas d'isomorphisme entre les processus binaires et les processus généraux. Nous avons une surjection : à chaque processus binaire on peut associer un processus général, mais le nombre de de processus binaires associés à chaque processus général est variable. Ainsi, du fait que nous nous intéressons à des valeurs typiques, c'est-à-dire en moyenne pour des processus de très grandes tailles, il est probable que ces moyennes soient biaisées suivant le modèle.

Pour un arbre syntaxique donné, il est classique (cf. l'article [CES86]) d'étudier sa correspondance sémantique. L'*arbre sémantique* associé à un processus représente l'ensemble de ses exécutions sous forme arborescente, avec un partage des préfixes communs. Avant de définir formellement l'arbre sémantique, rappelons quelques définitions classiques dans le cadre de structures arborescentes. Étant donné un arbre et ν l'un de ses nœuds. Les *enfants* de ν sont les nœuds descendants directement reliés à ν. L'*ensemble des descendants* de ν contient tous les enfants de ν, et récursivement l'ensemble des descendants de chacun de ses enfants. Le *sous-arbre* enraciné en ν est formé de ν et de l'ensemble de ses descendants. Nous considérons les arbres sémantiques comme des arbres enracinés, plans et d'arité quelconque. Leurs nœuds sont décorés via un nom d'action. Remarquons, dans la Figure 7.4 que le même nom d'action apparaît dans différents nœuds.

Afin de comprendre la structure de l'arbre sémantique associé à un processus général (plan), explicitons sa construction via les deux définitions suivantes.

Définition 7.1.4

Soit T un arbre syntaxique et $v_1, \dots, v_r$ les enfants de la racine. Pour $i \in \{1, \dots, r\}$, la i-contraction *de T est l'arbre plan, enraciné en v_i dont les descendants, ordonnés de gauche à droite, sont les sous-arbres $T(v_1), \dots, T(v_{i-1}), T(v_{i_1}), \dots, T(v_{i_m}), T(v_{i+1}), \dots, T(v_r)$, tel que $T(\nu)$ correspond au sous-arbre de T enraciné en ν et les sous-arbres $T(v_{i_1}), \dots, T(v_{i_m})$ sont les descendants de $T(v_i)$. Notons $T \lhd i$ la i-contraction de T.*

Si T est  alors $T \lhd 2$ est 

Nous remarquons, ici, que l'action a de la racine est remplacée par l'action de son second enfant c. Désormais, la construction de l'arbre sémantique suit la récurrence suivante.

Définition 7.1.5

Soit T un arbre syntaxique et $\textsc{Shuf}(T)$ *l'arbre sémantique associé, définit récursivement par :*

- *si T est une feuille, alors* $\textsc{Shuf}(T) = T$,
- *si T est enraciné en ν d'arité $r \geqslant 0$, alors* $\textsc{Shuf}(T)$ *est l'arbre plan enraciné en ν dont les sous-arbres enracinés en ses enfants sont, de gauche à droite* $\textsc{Shuf}(T \lhd 1), \dots, \textsc{Shuf}(T \lhd r)$.

Une exécution du processus T est la séquence d'actions stockée dans une branche de $\textsc{Shuf}(T)$ *(de la racine à une feuille).*

La construction de l'arbre sémantique associé à un processus binaire, est similaire à celle du modèle général. Cet arbre sémantique est (enraciné et plan) *d'arité quelconque*. En dehors de la racine, chaque nœud de l'arbre sémantique est décoré par une action. Si le processus global n'est pas un processus préfixé, comme dans notre exemple de la Figure 7.2, alors la racine de l'arbre sémantique n'est pas décorée : cf. la Figure 7.4.

Proposition 7.1.6

Pour le modèle de processus généraux, il y a un isomorphisme entre l'ensemble des arbres syntaxiques et l'ensemble des arbres sémantiques. Pour le modèle de processus binaires, de nombreux arbres syntaxiques sont

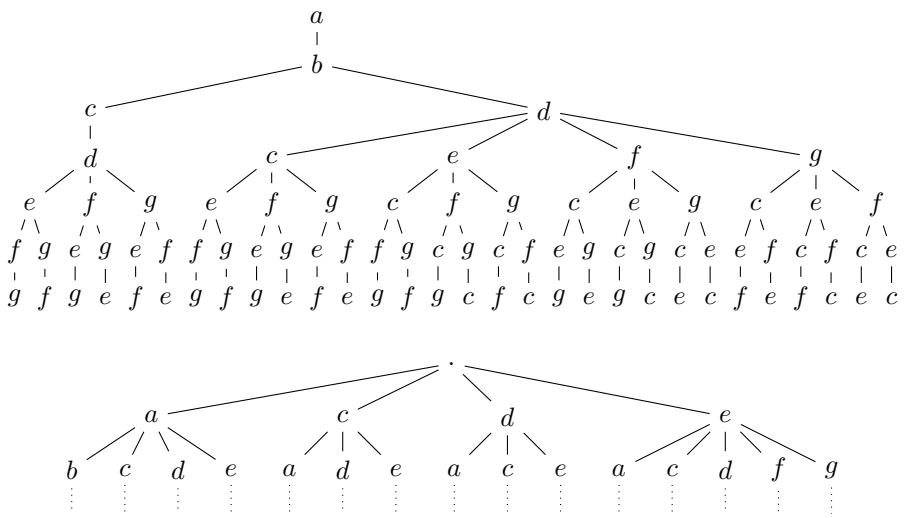


FIGURE 7.4 – *Les arbres sémantiques induit par le processus (général au dessus, les premiers niveaux du binaire en dessous).*

associés au même arbre sémantique.

En intégrant la *règle de commutativité* de l'opérateur parallèle, i.e. $P \parallel Q = Q \parallel P$, on passe naturellement des arbres syntaxiques plans aux arbres non plans. Ainsi, un nœud interne, a désormais un ensemble d'enfants plutôt qu'une séquence d'enfants. Cette adaptation, classique d'un point de vue combinatoire, se réfère aux travaux de Pólya [Pó37], et permet en particulier d'énumérer les structures arborescentes à symétrie près (par rapport aux enfants d'un nœud, et récursivement). Dans le livre de Flajolet et Sedgewick, ces derniers résultats sont intégrés au sein de la combinatoire analytique (par exemple [FS09, p. 475–477] pour les arbres généraux non plans). Dès lors que les arbres syntaxiques sont non-plans, il est naturel qu'il en soit de même pour les arbres sémantiques. Néanmoins, leurs caractéristiques qui nous intéressent ne sont pas impactées par le fait qu'ils soient considérés comme plans ou non.

Enfin, notons que dans le cas binaire, les nœuds internes binaires ont un ensemble de deux enfants au lieu d'une paire ordonnée. On se réfère aussi aux résultats établis par Pólya pour ce cas-là.

Les études quantitatives qui nous intéressent sont basées sur l'analyse de paramètres des arbres sémantiques. Il est commun de dire qu'on a affaire à une explosion combinatoire *lors du passage de l'arbre syntaxique au sémantique. Nous nous intéressons à quantifier cette explosion.*

7.1.1 Nombre moyen d'exécutions

La mesure fondamentale du nombre d'exécutions des processus est le fil conducteur de nos études quantitatives des modèles de plus en plus riches concernant les processus concurrents. Dans plusieurs modèles, nous nous sommes également intéressés au profil moyen des arbres sémantiques. En effet, cette mesure nous permet de calibrer convenablement des algorithmes aléatoires permettant d'explorer les arbres sémantiques jusqu'à une profondeur fixée, comme nous le verrons dans la Section 10.1.

Enfin rappelons que les résultats quantitatifs sont impactés par la planarité ou non d'un modèle. D'un point de vue technique, notons qu'en général, l'étude du modèle plan permet d'obtenir des expressions exactes concernant les résultats moyens. Mais dès que l'on s'intéresse aux modèles non plans, la théorie de Pólya [Pó37] concernant l'énumération des arbres permet d'obtenir des approximations sans pouvoir espérer de formules exactes.

Théorème 7.1.7

Soit $\bar{\mathcal{E}}_n$ le nombre moyen d'exécutions des processus parallèle de taille n. Il existe des constantes[2] *α et β telles que*

$$\bar{\mathcal{E}}_n \underset{n\to\infty}{\sim} \alpha\, \beta^n\, n!.$$

Le tableau de la Figure 7.5 présente les valeurs de α et β suivant que le modèle soit général ou binaire, plan ou non. Nous avons déjà noté que le nombre d'exécutions du

	général plan	général non-plan
α	$= 2$	$\approx \sqrt{n} \cdot 2.2731\ldots$
β	$= 1/2$	$\approx 0.33832\ldots$
	binaire plan	binaire non-plan
α	$\approx 1.1566\ldots$	$\approx 1.2431\ldots$
β	$\approx 0.79287\ldots$	$\approx 0.64156\ldots$

FIGURE 7.5 – *Valeurs numériques correspondantes au Théorème 7.1.7.*

processus où toutes les actions sont en parallèle est de l'ordre de $n!$. Nous remarquons toutefois que le nombre moyen d'exécutions est en dessous de cette valeur maximale, d'un facteur exponentiel. Mais même en moyenne,

2. Dans le Théorème 7.1.7, α et β sont des constantes, sauf dans le modèle général non plan.

le nombre de feuilles des arbres sémantiques explose par rapport à la taille des arbres syntaxiques.

Démonstration (idées) : L'idée fondamentale consiste à compter le nombre d'arbres syntaxiques croissants (avec les contraintes structurelles associées au modèle d'arbres). Intéressons-nous, par exemple, aux processus généraux plans. La classe combinatoire [3] $\mathcal{G}^{[c]}$ de ces arbres étiquetés croissants vérifie la spécification suivante

$$\mathcal{G}^{[c]} = \mathcal{Z}^{\square\star} \star \textsc{Seq}\, \mathcal{G}^{[c]}.$$

Le produit de Greene $^{\square\star}$ a été introduit dans la thèse [Gre83]. Il encode le fait que l'étiquette la plus petite de la structure est située dans la première composante du produit (réduite à $\mathcal{Z}$ ici). Récursivement, la spécification encode bien les arbres croissants (étiquetés de 1 à n). La méthode symbolique ([FS09, p. 139]) traduit la spécification en une équation intégrale vérifiée par la série génératrice exponentielle $G^{[c]}(z)$ associée à $\mathcal{G}^{[c]}$. Sa résolution et l'extraction du n-ième coefficient sont classiques. ■

Nous remarquons, en prenant en considération des modèles non-plans, que le nombre moyen d'exécutions est nettement inférieur. En effet, plus les actions sont en parallèle, plus les structures arborescentes sont symétriques et alors plus le nombre d'exécutions est grand. Néanmoins, bien que les quantités soient exponentiellement éloignées, les formes des asymptotiques sont relativement similaires, ce qui fait résonance à ce que Flajolet et Sedgewick ont écrit (cf. [FS09, p. 71–72]) :

> "[...] (some) universal law governs the singularities of simple tree generating functions, either plane or non-plane [...]".

On peut donc en conclure que les résultats sont plutôt robustes aux modifications induites par ces quatre modèles combinatoires, et donc le choix d'un modèle dont l'analyse est la plus simple, ne perturbe pas l'essence des résultats obtenus. De plus, les valeurs typiques obtenues par des moyennes arithmétique ou géométrique (nous savons que cette dernière est moins impactées par les cas extrémaux, cf. l'article [8]), ne sont pas non plus fondamentalement impactées par la combinatoire associée à la structure encodant la syntaxe des processus concurrents.

7.1.2 Profil moyen de l'arbre sémantique

En pratique, il pourrait être intéressant de construire entièrement l'arbre sémantique d'un processus donné. Néanmoins, bien rapidement, lorsque la taille du processus croît, on se rend compte que ce n'est pas envisageable. Est-il possible, efficacement, de calculer dans un premier temps le nombre de nœuds de l'arbre sémantique, et éventuellement sa répartition par niveau, i.e. son profil ?

La notion-clé lorsque l'on s'intéresse au profil des arbres sémantiques repose sur une structure extraite de l'arbre syntaxique. Cette structure particulière apparaît aussi dans un contexte plus algébrique [CK98], sous le nom de *coupe admissible*.

Définition 7.1.8

Soit T un arbre syntaxique de taille n. Une coupe admissible *de T, de taille $\ell \in \{1, \dots, n\}$ est obtenue en supprimant itérativement $n - \ell$ feuilles [4] de T.*
Une coupe admissible est dite croissante *si ses actions sont étiquetées de façon croissante.*

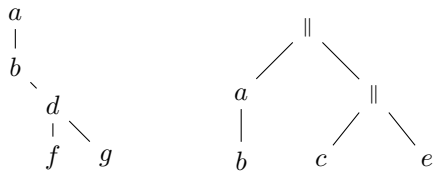


FIGURE 7.6 – *Une coupe admissible de chacun des processus des Figures 7.1 et 7.2.*

Avant de s'intéresser précisément au profil, énonçons un résultat concernant la taille globale de l'arbre sémantique. Commençons par les remarques suivantes : la racine d'une arbre est au niveau 0 ; pour un arbre syntaxique T (de taille n), le nombre de nœuds du niveau $\ell - 1$ de son sémantique correspond au cumul du nombre de coupes admissibles croissantes de taille ℓ de T (avec $\ell \in \{1, \dots, n\}$).

Théorème 7.1.9

Soit $\bar{\mathcal{E}}_n$ le nombre moyen d'exécutions des processus de taille n, soit $k \in \{0, \dots, n-1\}$ une constante, et $\bar{\mathcal{V}}_n^{n-1-k}$ le nombre moyen de nœuds au niveau $n-1-k$ des arbres sémantiques issus des processus de taille n. Pour les modèles de processus généraux plans, binaires plans ou non, on a

$$\bar{\mathcal{V}}_n^{n-1-k} \underset{n\to\infty}{\sim} \frac{\bar{\mathcal{E}}_n}{k!}.$$

Soit $\bar{\mathcal{V}}_n$ la taille moyenne des arbres sémantiques issus des processus de taille n. Pour les modèles de processus généraux plans et binaires, plans ou non, on a

$$\bar{\mathcal{V}}_n \underset{n\to\infty}{\sim} e \cdot \bar{\mathcal{E}}_n.$$

Dans un arbre sémantique, les deux derniers niveaux ont le même nombre de nœuds (lorsque l'on a exécuté l'avant dernière action, il n'en reste plus qu'une seule à exécuter). Ainsi, en moyenne, les deux derniers niveaux des arbres sémantiques contiennent $2 \cdot \bar{\mathcal{E}}_n$, c'est-à-dire une partie non négligeable de la taille globale.

Démonstration (idées) : L'idée fondamentale consiste à exhiber les coupes admissibles d'une taille fixée $n - k$ induites par les arbres syntaxiques de taille n (avec les contraintes structurelles associées au modèle d'arbres). Dès lors, il reste à les étiqueter de façon croissante afin

3. Lorsque $\mathcal{A}$ désigne une classe combinatoire de structures non étiquetées, la classe $\mathcal{A}^{[c]}$ désigne la classe des objets de $\mathcal{A}$ étiquetés de façon croissante.

4. Dans les modèles binaires, si la suppression d'une feuille rend un nœud ‖ sans aucun enfant ou avec un unique enfant, on élimine également ce nœud ‖.

d'obtenir le cumul du nombre de nœuds au niveau $n-1-k$ dans l'ensemble des arbres sémantiques issus des syntaxiques de taille n. ■

Sur la Figure 7.7, nous avons généré uniformément deux arbres de taille 40 (parmi tous les arbres généraux plans de taille 40). Nous avons représenté dans la même couleur que l'arbre syntaxique le profil de son sémantique (en échelle logarithmique). Le profil en rouge correspond au profil moyen sur l'ensemble des arbres de taille 40. On remarque que l'arbre syntaxique en bleu possède davantage d'actions en parallèle que celui en noir ; le profil en bleu est très proche du profil moyen, mais il reste en-deçà tout de même. La moyenne semble très fortement biaisée par les arbres où la plupart des actions sont en parallèle.

Expérimentalement, le Théorème 7.1.9 est également valide pour le modèle de processus généraux non-plans. La preuve n'a pas encore été écrite, mais l'approche simplifiée utilisée pour les processus binaires semblent être possible.

7.2 Parallélisme et choix non-déterministe

Dans l'article [19], au parallélisme, nous ajoutons un nouvel opérateur : le *choix non-déterministe*. Lorsque des processus sont précédés par l'opérateur de choix, exactement un seul d'entre eux sera exécuté. Complétons la Définition 7.1.1 pour définir les processus généraux plans non-déterministes.

Définition 7.2.1 (Processus non-déterministe)

- *une* action atomique, *notée* α, *est un processus, appelé* processus atomique,
- *un processus* P, *préfixé par une action* α, *noté* $\alpha.P$, *est un processus, appelé* processus préfixé,
- *la* composition en parallèle $P_1 \parallel \ldots \parallel P_n$ *d'un nombre fini de processus préfixés est un processus,*
- *le* choix non-déterministe $P_1 + P_2 + \ldots$ *d'au moins deux processus préfixés ou atomiques est un processus.*

Nous avons effectué l'analyse des processus généraux plans non-déterministes. De la même manière que précédemment, nous nous intéressons aux processus préfixés et aux processus dont la racine est un choix non-déterministe. Nous laissons de côté les processus dont la racine est une composition en parallèle. Par ailleurs, nous pourrions relâcher la contrainte de planarité et aussi s'intéresser à des opérateurs binaires. Mais, ce qui nous intéresse ici, c'est d'analyser l'évolution de l'explosion combinatoire lorsque nous ajoutons ce nouvel opérateur de choix. Le modèle général plan nous permet d'alléger la partie technique, tout en permettant la comparaison avec le modèle sans choix non-déterministe.

Intéressons-nous au processus exemple suivant

$$a.([(b.(c \parallel d)) + e] \parallel [f.((g + (h.(i \parallel j)) + k) \parallel l)]).$$

Sur la Figure 7.8, nous avons représenté l'arbre syntaxique du processus avec choix non-déterministe, ainsi que la tête de l'arbre sémantique associé.

Présentons une spécification nous permettant d'énumérer les arbres syntaxiques.

$$\begin{cases} \mathcal{C} &= \mathcal{C}_{\parallel} + \mathcal{C}_{+} \\ \mathcal{C}_{\parallel} &= \mathcal{Z} \times \textsc{Seq}\,(\mathcal{C}) \\ \mathcal{C}_{+} &= \mathcal{C}_{\parallel} \times \mathcal{C}_{\parallel} \times \textsc{Seq}(\mathcal{C}_{\parallel}) \end{cases}$$

La classe combinatoire $\mathcal{C}$ correspond à l'ensemble de tous les arbres syntaxiques. Elle se décompose en deux sous-classes : $\mathcal{C}_{\parallel}$ contient les processus atomiques ou préfixés et $\mathcal{C}_{+}$ contient les processus enracinés par un choix non-déterministe. La méthode symbolique nous permet de traduire directement le spécification précédente. En résolvant le système sur les séries génératrices, nous prouvons

$$C(z) = \frac{1}{2}\left(1 - z - \sqrt{1 - 6z + z^2}\right);$$

$$C_n \underset{n\to\infty}{\sim} \sqrt{\frac{3\sqrt{2}-4}{4\pi n^3}}\left(3 - 2\sqrt{2}\right)^{-n}.$$

Pour un processus P, un choix global de P contient l'ensemble des actions de P tel que si un sous-processus est un choix non-déterministe, seule une de ses branches apparaît dans le choix global. De façon plus formelle :

Définition 7.2.2

Soit P *un processus général plan avec choix non-déterministe. Un* choix global *de* P*, est un processus obtenu en ne conservant qu'un seul enfant des nœuds* $+$ *de choix non-déterministe.*

La Figure 7.9 présente un choix global Q issu du processus exemple. Nous remarquons désormais que, puisque les nœuds contenant l'opérateur $+$ ne comptent pas dans la taille du processus Q, ces nœuds peuvent être éliminés et on obtient dès lors un processus parallèle.

Définition 7.2.3

Soit P *un processus général plan avec choix non-déterministe. Une* exécution du processus non-déterministe P *est une exécution de l'un de ses choix globaux.*

Ainsi, $\langle a, b, f, l, d, c, g\rangle$ et $\langle a, b, f, l, d, g, c\rangle$ sont deux exécutions de l'exemple issu du même choix global. L'exécution $\langle a, e, f, h, i, j, l\rangle$ est issue d'un autre choix global.

De par le comportement du choix non-déterministe, nous observons immédiatement que, dans les arbres sémantiques, en particulier celui de la Figure 7.8, désormais les chemins partant de la racine et allant jusqu'à une feuille dans l'arbre sémantique ne contiennent plus tous les mêmes actions (et ne sont pas forcément de la même longueur). De façon immédiate, on comprend que le nouvel opérateur de choix réduit l'explosion combinatoire, puisque toute action du processus n'est pas forcément exécutée.

Finalement, une exécution d'un processus non-déterministe est l'étiquetage croissant des actions de l'un

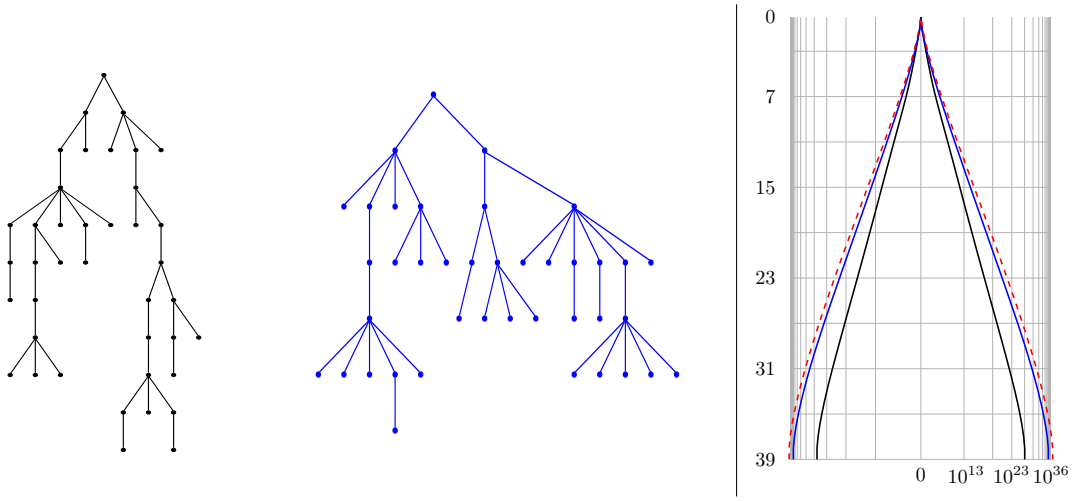


FIGURE 7.7 – *Deux processus généraux plans de taille* 40 *et les profils de leurs sémantiques.*

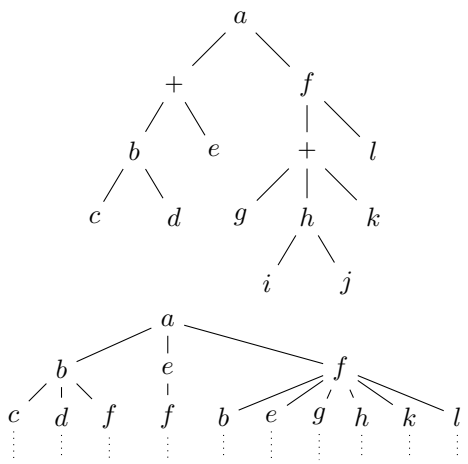


FIGURE 7.8 – *L'arbre syntaxique d'un processus non-déterministe et les trois premiers niveaux de son arbre sémantique.*

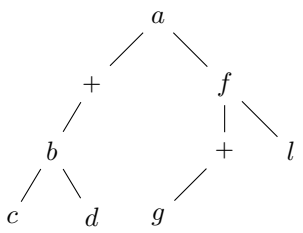


FIGURE 7.9 – *Un choix global du processus de la Figure 7.8.*

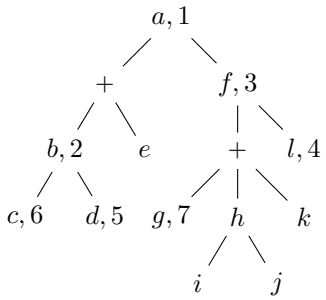


FIGURE 7.10 – *Exécution* $\langle a, b, f, l, d, c, g\rangle$.

de ses choix globaux. On appellera aussi cet étiquetage d'un choix global, *étiquetage partiel croissant* du processus non-déterministe. Sur la Figure 7.10, on a représenté une exécution du processus avec choix non-déterministe sous la forme d'un étiquetage partiel croissant du processus de la Figure 7.8. On obtient donc l'analogue de la Proposition 7.1.3.

Proposition 7.2.4

Soit P un processus général plan avec choix non-déterministe. Le nombre d'exécutions de P est égal au nombre d'étiquetages partiels croissants (des actions) de son arbre syntaxique.

Afin d'énumérer la classe associées au cumul du nombre d'exécutions des processus (de même taille), nous procédons comme auparavant : nous introduisons une seconde variable qui marque les contraintes de croissance de certaines actions.

$$\begin{cases} \mathcal{C}^{[c]} &= \mathcal{C}_{\|}^{[c]} + \mathcal{C}_{+}^{[c]} \\ \mathcal{C}_{\|}^{[c]} &= \mathcal{U}^{\square} \star \mathcal{Z} \times \textsc{Seq}\,\mathcal{C}^{[c]} \\ \mathcal{C}_{+}^{[c]} &= \mathcal{C}_{\|}^{[c]} \times \mathcal{C}_{\|} \times \textsc{Seq}\,\mathcal{C}_{\|} + \mathcal{C}_{\|} \times \textsc{Seq}\,\mathcal{C}_{\|} \times \mathcal{C}_{\|}^{[c]} \times \textsc{Seq}\,\mathcal{C}_{\|}. \end{cases}$$

La marque $\mathcal{Z}$ est dédiée à chaque action alors que $\mathcal{U}$ ne marque que les actions étiquetées. Dans le contexte de la série bivariée $C^{[c]}(z,u)$, la série est ordinaire en z et exponentielle en u.

Théorème 7.2.5

Soit $\bar{\mathcal{E}}_n$ le nombre moyen d'exécutions des processus parallèle avec choix non-déterministe de taille n. Il existe

des constantes α et β telles que

$$\bar{\mathcal{E}}_n \underset{n\to\infty}{\sim} \alpha\, \beta^n\, n!,$$

avec $\alpha \approx 5.5183\ldots$ et $\beta \approx 0.34314\ldots$.

Ce résultat est à comparer avec son analogue dans le modèle général plan, sans choix non-déterministe (cf. le Théorème 7.1.7), où nous montrons que $\bar{\mathcal{E}}_n \underset{n\to\infty}{\sim} n!/2^{n-1}$. Dans les arbres typiques, l'opérateur de choix non-déterministe a un effet de coupe non négligeable par rapport à l'explosion combinatoire.

Chapitre 8

Parallélisme et Synchronisation

Au parallélisme, nous ajoutons un nouveau concept : la *synchronisation*. D'un point de vue concurrence, voilà une définition informelle de la synchronisation. Supposons que l'on ait trois processus P_1, P_2 et P_3. Les deux premiers sont en parallèle ; le troisième ne peut être exécuté que lorsque les deux premiers ont été *entièrement* exécutés. Afin de contraindre ce comportement, à la suite des processus P_1 et P_2, on ajoute une action dite de synchronisation. Celle-ci sera suivie du processus P_3.

Il est à noter que nous avons choisi de définir les processus parallèle avec synchronisation, tels qu'ils puissent être représentés via un graphe dirigé acyclique (DAG). Ainsi, nous supposons que les synchronisations d'un processus peuvent être définies statiquement et non dynamiquement. De plus, nous introduisons la notion d'action muette, qui permet d'éviter la représentation des opérateurs au sein des DAG, comme dans le modèle de processus généraux.

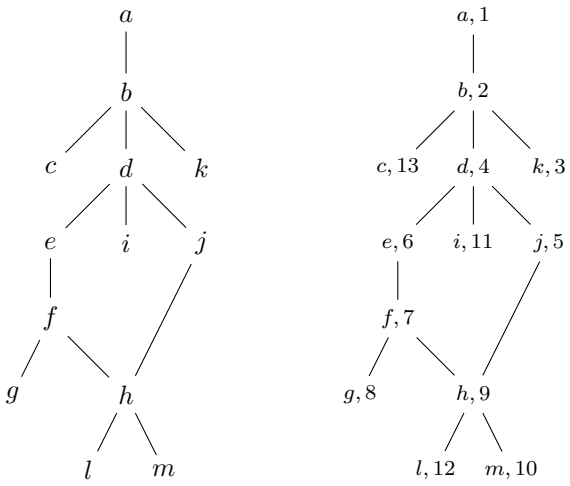


FIGURE 8.1 – *Un processus général avec une synchronisation h.*

8.1 Une unique synchronisation

Commençons par présenter un modèle simplifié : dans un processus parallèle, nous ajoutons exactement une synchronisation. Ce modèle a été présenté dans l'article [10]. Présentons un premier exemple du processus parallèle ayant une synchronisation. Il s'agit d'une extension des processus généraux plans. Dans le processus représenté à gauche sur la Figure 8.1, avant de pouvoir exécuter l'action h (donc aussi avant l et m), il faut avoir exécuté les actions f et j. Par contre l'ordre d'exécution de g et h n'est pas contraint. Sur la droite, on a représenté une exécution valide $\langle a, b, k, d, j, e, f, g, h, m, i, l, c\rangle$.

D'un point de vue structurel, on peut voir ce modèle comme un graphe acyclique dirigé avec une contrainte forte : on a une unique source (que nous appelons la racine) ; un unique nœud avec un degré entrant 2, les autres nœuds ayant un degré entrant 1. Dès lors, une exécution du processus correspond à un étiquetage croissant de la structure de graphe.

Cependant, en dupliquant le nœud h, pour séparer les enfants de f et ceux de j et en choisissant par exemple de définir les descendant de h comme des descendants de f, on obtient à nouveau une structure arborescente particulière (cf. la partie gauche de la Figure 8.2). Ce type d'arbre

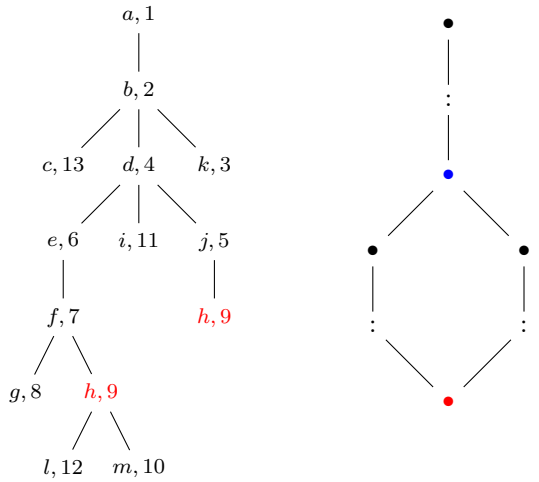


FIGURE 8.2 – *(à gauche) Un processus général avec une synchronisation, représenté sous forme arborescente ; (à droite) Le squelette d'un processus avec une synchronisation.*

étiqueté est très intéressant d'un point de vue combinatoire. Il s'agit d'une sorte d'arbre croissant avec une répétition. Dans l'article [10] nous avons considéré exactement l'arbre général plan croissant avec une étiquette répétée. Dans la Section 11.1, nous poursuivons ce type de perspective naturelle d'arbres croissants avec des répétitions d'étiquettes.

Toutefois, dans ce mémoire, pour satisfaire davantage notre contexte de concurrence, nous ajoutons également une contrainte sur le deuxième nœud contenant l'étiquette répétée (dans le parcours préfixe) : il ne contient pas d'enfant. Cette contrainte a été laissée de côté dans l'article originel [10]. Néanmoins, l'adaptation est directe et l'analyse est complètement analogue. À première vue, les résultats ne devraient pas être très éloignés de ceux concernant les processus parallèle. En effet, seule une action est un peu particulière.

Afin de spécifier aisément le problème, nous considérons le squelette représenté à droite sur la Figure 8.2. Le squelette est constitué d'une suite (éventuellement vide) de nœuds noirs suivie d'un nœud bleu. Ce dernier est suivi de deux suites (éventuellement vides) de nœuds noirs qui se synchronisent au niveau du nœud rouge. Afin de construire un processus général avec une synchronisation, il suffit de substituer chaque type de nœud :

- remplacé par un nœud racine de 3 suites d'arbres généraux plans
- remplacé par un nœud racine de 3 suites d'arbres généraux plans
- remplacé par un nœud racine d'1 suite d'arbres généraux plans.

Proposition 8.1.1

Soit $\mathcal{S}$ la classe des processus généraux plans ayant une synchronisation. La série génératrice ordinaire $S(z)$ associée est telle que

$$S(z) = \frac{z}{2}\frac{\left(1-\sqrt{1-4z}\right)^4}{\left(4z-1+\sqrt{1-4z}\right)^3}$$

$$S_n \underset{n\to\infty}{\sim} \sqrt{\frac{n}{\pi}}\, 2^{n-2}.$$

Le passage aux structures croissantes, peut se faire, dans ce cas simple avec l'opérateur de Greene, même si nous avons présenté une méthode plus élaborée, et plus générale dans l'article [10]).

Théorème 8.1.2

Soit $\bar{\mathcal{E}}_n$ le nombre moyen d'exécutions des processus parallèle avec une synchronisation. On a

$$\bar{\mathcal{E}}_n \underset{n\to\infty}{\sim} \frac{4}{3}\, 2^{-n}\, n!.$$

En dehors de la constante multiplicative, nous retrouvons le même résultat que pour les processus parallèle, généraux et plans, ce qui semble raisonnable par rapport à la modification minime dans les modèles.

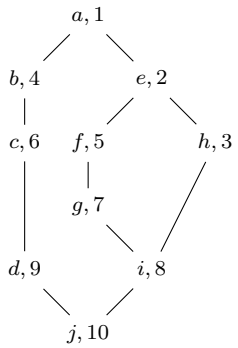


FIGURE 8.3 – *Un diamant binaire croissant, de source a et de puits j.*

8.2 DAG diamants

Afin d'augmenter l'expressivité du modèle, nous souhaitons rendre possible un nombre quelconque de synchronisations, sans toutefois tomber dans le cas d'un processus quelconque.

Une première approche consiste à construire récursivement des objets de façon plutôt simple. Cette direction peut être suivie en se basant sur la structure de *diamant* que nous caractérisons informellement de la manière suivante. Un diamant est construit à partir de deux nœuds particuliers : une source et un puits, tous deux liés à un certain nombre de sous-structures qui sont des diamants. Nous avons introduits cette structure de DAG croissants dans notre article [24] qui fait écho à un article classique de la littérature [BFS92] sur les familles d'arbres croissants. Bien que les structures d'arbres croissants soient fondamentalement distinctes, la manière de les étudier quantitativement et d'analyser les propriétés typiques sont très similaires. Ces familles de structures croissantes sont aussi caractérisées par des équations différentielles non linéaires d'ordre deux, qui ont été étudiées notamment dans l'articles [KP12].

Une famille de diamants est définie via la spécification suivante

$$\mathcal{D} = \mathcal{Z} + \mathcal{Z} \times G(\mathcal{D}) \times \mathcal{Z}.$$

La fonction G est la fonction qui définit les degrés de la famille de diamants : binaires, ternaires, généraux, plan ou non, par exemple. Notons que cette structure non étiquetée peut être vue comme une famille d'arbres. Mais, lors de l'étiquetage croissant, c'est-à-dire pour l'énumération des exécutions des diamants, on obtient la spécification suivante, bien éloignée de la structure d'arbre croissant.

$$\mathcal{D}^{[c]} = \mathcal{Z} + \mathcal{Z}\ {}^{\square}\star\ G(\mathcal{D}^{[c]})\ \star^{\blacksquare}\ \mathcal{Z}.$$

Les deux opérateurs $\star^{\square}$ et $\star^{\blacksquare}$ sont les désormais classiques produits de Greene [Gre83], signifiant, dans cette équation, respectivement que le premier atome contient l'étiquette la plus petite et le second, l'étiquette la plus grande. Ces opérateurs ont été adaptés à la combinatoire

analytique et sont présentés de façon détaillée [FS09, Section II.6.3.].

Si nous nous restreignons au cas binaire, par exemple $G(x) = 1 + x + x^2$, on se retrouve dans le contexte des fonctions elliptiques de Weierstrass [AS12], pour lesquelles nous avons même la possibilité d'exhiber une formule exacte du nombre de diamants croissants de taille n. Nous avons représenté un diamant croissant binaire sur la Figure 8.3.

Théorème 8.2.1

Soit $\bar{\mathcal{E}}_n$ le nombre moyen d'exécutions des processus binaires plans. On a

$$\bar{\mathcal{E}}_n \underset{n\to\infty}{\sim} \alpha\,\sqrt{n}\,\beta^n\,n!,$$

avec $\alpha \approx 0.56633\ldots$ et $\beta \approx 0.67159\ldots$.

En comparaison avec le modèle d'arbres parallèles binaires, le fait d'avoir des synchronisations diminue la partie exponentielle de croissance (mais la partie factorielle reste inchangée).

Parmi les différentes familles de diamants que l'on peut construire, le modèle des diamants généraux plans se comporte de façon non classique. En voilà les spécification (les diamants et leurs exécutions) :

$$\mathcal{D} = \mathcal{Z} + \mathcal{Z} \times \textsc{Seq}(\mathcal{D}) \times \mathcal{Z},$$
$$\mathcal{D}^{[c]} = \mathcal{Z} + \mathcal{Z}\ {}^{\square}\star\ \textsc{Seq}(\mathcal{D}^{[c]})\ \star\ \mathcal{Z}\ {}^{\blacksquare}.$$

En effet, comme prouvé dans l'article [24], l'équivalent asymptotique du nombre de diamants croissants se comporte ainsi

$$\frac{n!\,\rho^{1-n}}{n^2\sqrt{2\log n}}\left(\sum_{0\leqslant k<K}\frac{P_k(\log\log n)}{(\log n)^k}+\mathcal{O}\left(\frac{(\log\log n)^K}{(\log n)^K}\right)\right).$$

Par ailleurs, dans la version longue de l'article, en préparation, nous prouvons que l'arité moyenne de la racine et la longueur de cheminement moyenne, lorsque les diamants sont grands, se comportent de façon non conventionnelles. Les diamants croissants typiques de ce modèle sont très écrasés : leur hauteur est très faible. Ainsi, en dehors de notre contexte de la concurrence, ce modèle de graphes pourrait avoir des application pour modéliser des graphes avec quelques nœuds de très forts degrés.

8.3 DAG série-parallèle

La sous-classe des diamants est trop contrainte, dans le sens où les mises en parallèle et synchronisation associées ne sont que possibles récursivement dans une sous-structure.

Nous allons nous concentrer sur l'équivalent de la sous-classe de posets série-parallèle dans le contexte des processus. Cette classe est très intéressante pour plusieurs raisons. Tout d'abord, d'un point de vue structurel, de nombreux problèmes difficiles dans les graphes généraux peuvent être résolus en temps raisonnable dans les graphes série-parallèle. Un certain nombre de résultats combinatoires sur les posets série-parallèle sont présentés par Stanley dans son livre [Sta86, chapitre 3]. Nos résultats dans le contexte des processus concurrents ont un écho direct pour la combinatoire des ordres partiels.

Nous choisissons de restreindre légèrement les DAG série-parallèle auxquels nous nous intéressons aux DAG binaires, qui correspondent à un ensemble classique de processus : les *processus Fork/Join* qui rappellent le fonctionnement des processus concurrents au sein des systèmes Linux. Cette contrainte d'opérateurs binaires ne modifie pas les difficultés techniques d'analyse (cf. notre article [26] justifiant ce point) comme cela a été le cas dans les processus purement parallèle. Par ailleurs, du point de vue algorithmique, nous y prouvons aussi que la restriction binaire peut être adaptée de façon directe pour s'appliquer aux processus série-parallèle généraux. Informellement, un processus $\mathcal{FJ}$ est structuré tel la spécification représentée sur la Figure 8.4.

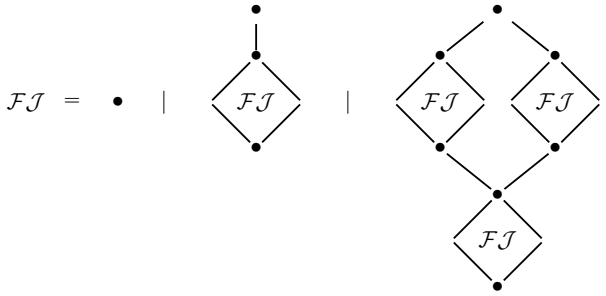


FIGURE 8.4 – *Les processus $\mathcal{FJ}$.*

Chaque nœud qui est suivi par deux sous-processus en parallèle est appelé *nœud Fork* (par exemple a, b et h sur la Figure 8.5) et est associé à une synchronisation, appelée *nœud Join* (respectivement les nœuds h, g et k).

Définition 8.3.1

La spécification des processus $\mathcal{FJ}$ est

$$\mathcal{FJ} = \bullet \mid \bullet \times \mathcal{FJ} \mid \bullet \times (\mathcal{FJ} \times \mathcal{FJ}) \times \mathcal{FJ}.$$

La spécification de la classe $\mathcal{FJ}$ laisse envisager un isomorphisme avec une certaine classe arborescente, les arbres dont les nœuds sont d'arité $0, 1$ ou 3. La Figure 8.5 présente un isomorphisme possible. Il s'agit d'une famille d'arbres simplement générés (cf. [FS09]). L'analyse quantitative de tels arbres (non étiquetés) est classique.

Proposition 8.3.2

Soit $F(z)$ la série génératrice de la classe $\mathcal{FJ}$. On a

$$F(z) = -2\sqrt{\frac{1-z}{3z}}\,\sin\left(\frac{\pi}{6}+\frac{2}{3}\arctan\left(\frac{2\sqrt{1-3z+3z^2-\frac{31}{4}z^3}}{(3z)^{3/2}-2(1-z)^{3/2}}\right)\right).$$

Le nombre de processus $\mathcal{FJ}$ de taille n est asymptoti-

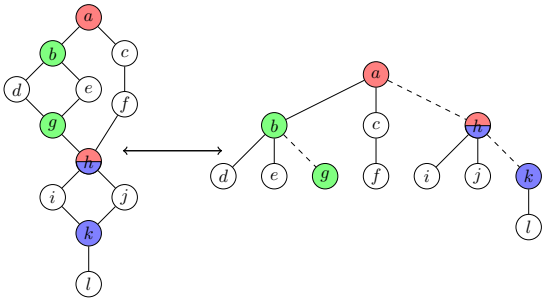


FIGURE 8.5 – *Les processus $\mathcal{FJ}$ encodés par des arbres ternaires.*

quement,

$$F_n \underset{n\to\infty}{\sim} \frac{1}{2}\sqrt{\frac{1}{3}+\frac{1}{2^{2/3}}}\frac{\rho^{-n}}{\sqrt{\pi n^3}},$$

avec la singularité dominante $\rho = (1+3\cdot 2^{-2/3})^{-1}$.

Si nous souhaitons désormais nous intéresser aux exécutions des processus $\mathcal{FJ}$, c'est-à-dire aux structures de la classe $\mathcal{FJ}$, que nous étiquetons de façon croissante, nous nous rendons rapidement compte que le produit de Greene ne permet pas de gérer efficacement cet étiquetage, aussi bien pour la mise en parallèle que la mise en série de plusieurs sous-processus : cf. [10] mais également [26].

8.3.1 Produit ordonné

Dès la section précédente, relative aux processus avec une synchronisation, nous nous sommes aperçus que le produit contraint de Greene n'était pas le mieux adapté à notre contexte. Nous pourrions bien entendu ré-encoder nos problèmes de croissances avec le produit de Greene, mais la spécification contiendrait probablement autant de termes qu'il y a d'exécutions à compter. En effet, il faudrait en quelque sorte gérer l'entrelacement explicitement dans la spécification, qui naturellement deviendrait de taille exponentielle. Il est donc évident que cette approche ne peut être envisagée.

Voilà l'approche que nous avons présentée dans l'article [26]. Il nous faut définir un produit qui permette d'encoder une contrainte plus forte et plus globale. En effet, sur l'exemple de la Figure 8.5, toute exécution est telle que les étiquettes des actions de a à g sont toutes plus petites que les étiquettes des actions de h à l. Dans le contexte des espèces combinatoires (cf. [BLL98]), le *produit ordonné* a été introduit afin d'encoder une telle contrainte globale de croissance. Ce produit a aussi été utilisé dans un contexte algorithmique dans l'article [DPRS12]. Nos besoins concernant ce produit sont de l'ordre quantitatifs, et nous avons donc développé une analyse combinatoire du produit avant de l'utiliser pour spécifier notre modèle.

Définition 8.3.3

Soit $\mathcal{A}$ et $\mathcal{B}$ deux classes combinatoires étiquetées, et α et β deux structures respectivement dans $\mathcal{A}$ et $\mathcal{B}$. La classe de structures étiquetées induites par α et β est

$$\alpha \boxtimes \beta = \big\{(\alpha, f_{|\alpha|}(\beta)) \mid f_{|\alpha|}(\cdot) \text{ translate les étiquettes de l'argument de } +|\alpha|\big\}.$$

On étend naturellement le produit ordonné *aux classes combinatoires*

$$\mathcal{A} \boxtimes \mathcal{B} = \bigcup_{\alpha\in\mathcal{A},\ \beta\in\mathcal{B}} \alpha \boxtimes \beta.$$

En particulier, le produit ordonné $\mathcal{A} \boxtimes \mathcal{B}$ contient les éléments du produit $\mathcal{A} \star \mathcal{B}$ tels que toutes les étiquettes de la composante issue de $\mathcal{A}$ sont plus petites que toutes les étiquettes de la composante issue de $\mathcal{B}$.

Remarquons que dans le cas où l'une des deux classes du produit ordonné est réduite à un seul atome $\mathcal{Z}$, le produit ordonné correspond à l'un des produits classiques de Greene. Nous sommes désormais en mesure de spécifier les processus $\mathcal{FJ}$ étiquetés de façon croissante.

Proposition 8.3.4

Soit $\mathcal{FJ}^{[c]}$ la classe des processus $\mathcal{FJ}$ croissants.

$$\mathcal{FJ}^{[c]} = \mathcal{Z} + \mathcal{Z} \,\square\star\, \mathcal{FJ}^{[c]} + \mathcal{Z} \,\square\star\, \Big(\Big(\mathcal{FJ}^{[c]} \star \mathcal{FJ}^{[c]}\Big) \boxtimes \mathcal{FJ}^{[c]}\Big).$$

Nous avons les outils pour traduire de façon directe une telle spécification basée sur le produit ordonné dans le contexte des séries génératrices associées. Pour ce faire, rappelons deux transformées classiques : la transformée de Laplace combinatoire et la transformée de Borel combinatoire. Elles agissent ainsi sur les séries :

$$\mathcal{L}_c\left(\sum_{n\geqslant 0} a_n \frac{z^n}{n!}\right) = \sum_{n\geqslant 0} a_n z^n; \quad \mathcal{B}_c\left(\sum_{n\geqslant 0} a_n z^n\right) = \sum_{n\geqslant 0} a_n \frac{z^n}{n!}.$$

De manière analogue aux transformées de Laplace traditionnelles, le produit de deux transformées peut être écrit sous la forme d'un produit de convolution :

$$\mathcal{L}_c\, f \cdot \mathcal{L}_c\, g = \mathcal{L}_c\left(\int_0^z f(t)g'(z-t)dt + g(0)f(z)\right).$$

Notons la convolution combinatoire

$$f * g\,(z) = \int_0^z f(t)g'(z-t)\mathrm{d}t + g(0)f(z).$$

Proposition 8.3.5

Soit $\mathcal{A}$ et $\mathcal{B}$ deux classes combinatoires étiquetées. La fonctions génératrice exponentielle $C(z)$, associée à la classe $\mathcal{C} = \mathcal{A} \boxtimes \mathcal{B}$, satisfait les trois formes suivantes :

$$C(z) = \mathcal{B}_c\left(\mathcal{L}_c\, A(z) \cdot \mathcal{L}_c\, B(z)\right) = \sum_{n\geqslant 0} \frac{\sum_{k=0}^n a_k b_{n-k}}{n!} z^n$$
$$= A(z) * B(z).$$

De manière immédiate, on prouve que l'holonomie est stable pour le produit ordonnée puisqu'elle est stable pour

chacune des deux transformées combinatoires (et aussi pour le produit).

La proposition précédente nous permet de traduire la spécification des processus $\mathcal{FJ}$ croissants

$$F^{[c]}(z) = z + \int_0^z F^{[c]}(t)\ \mathrm{d}t + \int_0^z \int_0^t {F^{[c]}}^2(u) {F^{[c]}}'(t-u)\ \mathrm{d}u\ \mathrm{d}t.$$

L'étude de la série $F^{[c]}(z)$ semble difficile. Afin de pouvoir obtenir des résultats quantitatifs nous avons rajouté des contraintes sur la classe $\mathcal{FJ}$.

8.3.2 Sous-classes des processus Fork/Join

La première sous-classe qui nous intéresse consiste à contraindre le nombre d'opérateurs parallèle imbriqués. Voilà la définition formelle (de la classe étiquetée croissante) :

$$\begin{cases} \mathcal{W}_0^{[c]} & = \textsc{Set}_{\geqslant 1}\, \mathcal{Z}, \\ \mathcal{W}_\ell^{[c]} & = \mathcal{Z} + \mathcal{Z}^{\,\square} \star \mathcal{W}_\ell^{[c]} \\ & \quad + \mathcal{Z}^{\,\square} \star \left(\left(2 \cdot \mathcal{W}_{\ell-1}^{[c]} \star \textsc{Set}\, \mathcal{Z} - \mathcal{W}_0^{[c]} \star \mathcal{W}_0^{[c]} \right) \boxtimes \mathcal{W}_\ell^{[c]} \right). \end{cases}$$

Ces sous-classes de processus $\mathcal{FJ}$ représentent une séquence croissante de sous-classes, dans le sens $\mathcal{W}_0 \subset \mathcal{W}_1 \subset \mathcal{W}_2 \subset \dots$. Il y a deux restrictions majeures par rapport à processus $\mathcal{FJ}$, toutes deux associées aux nœuds Fork. La première est telle que $\mathcal{W}_\ell$ ne permet d'avoir que ℓ nœuds Fork imbriqués. La seconde est telle que deux processus ne peuvent être mis en parallèle que si l'un d'entre eux (au moins) est contraint à être une séquence d'atomes (un fil). Il s'agit du moyen le plus simple pour faire du parallélisme en concurrence. Par ailleurs, d'un point de vue technique, la série génératrice satisfait une propriété intéressante.

Proposition 8.3.6

Soit $\ell \in \mathbb{N}$. La classe $\mathcal{W}_\ell^{[c]}$ est telle que la fonction $\mathcal{L}_{\mathrm{c}}(W_\ell^{[c]}(z))$ est une fonction rationnelle, i.e. $W_\ell^{[c]}(z)$ satisfait une équation différentielle linéaire.

Afin de prouver cette dernière proposition, nous avons introduit un nouveau produit, le produit coloré, qui est en quelque sorte le dual du produit ordonné.

Remarquons que puisque $W_\ell^{[c]}(z)$ satisfait une équation différentielle linéaire, alors la série est holonome.

Proposition 8.3.7

Le nombre moyen d'exécutions des processus de $\mathcal{W}_{20}$ de taille n vérifie $\Theta(r^n)$, avec $r \approx 5.4314\dots$.

Nous pouvons augmenter l'expressivité du modèle en considérant la spécification suivante :

$$\begin{cases} \mathcal{N}_0^{[c]} & = \textsc{Set}_{\geqslant 1}\, \mathcal{Z}, \\ \mathcal{N}_\ell^{[c]} & = \mathcal{Z} + \mathcal{Z}^{\,\square} \star \mathcal{N}_\ell^{[c]} + \mathcal{Z}^{\,\square} \star \left(\left(\mathcal{N}_{\ell-1}^{[c]} \star \mathcal{N}_{\ell-1}^{[c]} \right) \boxtimes \mathcal{N}_\ell^{[c]} \right). \end{cases}$$

La seule distinction entre la classe $\mathcal{N}_\ell$ et $\mathcal{FJ}$ repose sur le nombre de nœuds Fork imbriqués. Il est borné par ℓ pour la classe $\mathcal{N}_\ell$ (i.e. au plus 2^ℓ processus peuvent être mis en parallèle). De façon analogue à la classe $\mathcal{W}_\ell$, nous prouvons que $\mathcal{N}_\ell^{[c]}$ vérifie une équation différentielle linéaire, mais dont l'ordre d_ℓ croit très rapidement quand ℓ augmente. Ainsi nous avons

$$d_1 = 3; \qquad d_\ell = \frac{(d_{\ell-1}+1)(d_{\ell-1}+2)}{2}.$$

En conséquence, la classe $\mathcal{N}_4$ semble être la limite permettant d'obtenir des résultat quantitatifs.

Alors que dans les premières sous-classes de processus nous avons étudié une limitation sur la profondeur de parallélisme, pour la sous-classe suivante nous allons nous intéresser à la profondeur de série.

Nous pouvons désormais construire des séquences de diamants. L'opérateur $\textsc{Set}^{\boxtimes}$ correspond à une séquence d'opérateurs $\boxtimes$, introduite dans l'article [26].

Théorème 8.3.8

Soit $\mathcal{S}^{[c]} = \textsc{Set}^{\boxtimes}\, \mathcal{D}^{[c]}$ la classe des séquences de diamants croissants. L'asymptotique du nombre de structures $S_n^{[c]}$ est équivalent à l'asymptotique du nombre de diamants croissants :

$$[z^n] S_n^{[c]} \underset{n\to\infty}{\sim} [z^n] D_n^{[c]}.$$

Le nombre moyen d'exécutions $\bar{\mathcal{E}}_n$ dans les séquences de diamants de taille n satisfait

$$\bar{\mathcal{E}}_n \underset{n\to\infty}{\sim} \alpha\ \beta^{n+1}\ (n+1)!$$

avec $\alpha \approx 15.704\dots$ et $\beta \approx 0.13690\dots$.

Concluons cette section en revenant sur la classe globale des processus $\mathcal{FJ}$. Tout d'abord, nous remarquons que la classe limite $\mathcal{N}_\ell$ (quand ℓ tend vers l'infini) correspond à la classe $\mathcal{FJ}$. Néanmoins, il est probable que la propriété d'holonomie soit perdue pour $\mathcal{FJ}$. En étudiant directement la suite $(\mathcal{FJ}_n^{[c]})$, nous sommes en mesure de prouver le théorème suivant.

Théorème 8.3.9

Le nombre d'exécutions moyen des processus $\mathcal{FJ}$ de taille n vérifie asymptotiquement, lorsque n tend vers l'infini

$$\alpha_1\ \beta_1^n\ (n+1)! \lesssim \bar{\mathcal{E}}_n \lesssim \alpha_2\ \beta_2^n\ (n+1)!,$$

avec pour la borne inférieure $\alpha_1 \approx 2.4078\dots$ et $\beta_1 \approx 0.11534\dots$ et pour la borne supérieure $\alpha_2 \approx 9.6315\dots$ et $\beta_2 \approx 0.23068\dots$.

Les croissances exponentielles des deux bornes sont distinctes. Pour obtenir ces bornes, une récurrence simple, vraie à partir de $n = 3$ montre que, asymptotiquement, la croissance exponentielle de $FJ_n^{[c]}/(n+1)!$ est comprise entre $1/3$ et $2/3$. Nous sommes en train de terminer la preuve du résultat suivant.

Conjecture 8.3.10

Le nombre d'exécutions moyen des processus $\mathcal{FJ}$ de

taille n vérifie asymptotiquement,

$$\bar{\mathcal{E}}_n \underset{n\to\infty}{\sim} \frac{12\sqrt{\pi}\, n^{3/2}}{\eta^2\sqrt{\frac{1}{3}+\frac{1}{2^{2/3}}}} \left(\frac{\rho}{\eta}\right)^n (n+1)!,$$

avec $\dfrac{\rho}{\eta} \approx 0.14967\ldots$.

Au final, les structures sont simplement relâchées aux fur et à mesure, pour s'approcher obtenir le modèle général. Les analyses quantitatives sont de plus en plus techniques à obtenir mais on aperçoit au final que les évolutions des modèles transparaissent dans les résultats quantitatifs, sans en modifier de façon flagrante les valeurs.

Chapitre 9

Comptage des exécutions

Les problèmes quantitatifs qui nous ont intéressés auparavant concernaient des résultats en moyenne pour des classes de processus. Dans les deux derniers chapitres, nous nous intéressons à des algorithmes effectifs pour des processus fixés. Le premier problème que nous souhaitons aborder est le comptage des exécutions d'un processus.

En se concentrant sur le modèle de processus parallèle avec multiples synchronisations, nous nous heurtons à la limite théorique [BW91] prouvée par Brightwell et Winkler. Ces-derniers ont démontré, dans le contexte équivalent des ordres partiels, que l'analogue du comptage du nombre d'exécutions d'un processus est $\#P$-complet. Rappelons que ce problème n'est pas directement relié aux comptages en moyenne que nous effectuons depuis le début de la partie sur la concurrence. Néanmoins, nous considérons que ces comptages en moyenne ne constituent que les étapes initiales qui nous permettent dans certains cas d'affiner la compréhension de la difficulté du comptage exact. Plutôt que d'attaquer le problème général de front, nous avons exhibé des classes naturelles d'un point de vue concurrence, pour lesquelles nous avons présenté des analyses quantitatives. Nous allons dans ce chapitre introduire des algorithmes de comptage d'exécutions efficaces.

Dans le cadre des processus parallèle, une formule classique de comptage nous permet d'obtenir, pour un processus donné, le nombre de ses exécutions. La *formule des équerres* [Knu98, p. 67] associée à une structure arborescente permet de compter le nombre d'étiquetages croissants de cette structure. Rappelons la formule dans le cas des arbres généraux, qui est complètement adaptée aux processus avec pour unique opérateur le parallélisme

Fait 9.0.1

Soit P l'arbre syntaxique d'un processus général (plan ou non). Le nombre ℓ_P d'exécutions du processus vaut

$$\ell_P = \frac{|P|!}{\prod_{S \text{ sous-arbre de } P} |P|}.$$

Il est immédiat qu'un algorithme de comptage peut être écrit dont la complexité en nombre d'opérations arithmétiques est linéaire en le nombre d'actions du processus. Il opère en un seul parcours de l'arbre.

9.1 Processus Fork/Join

Nous avons établi une formule des équerres pour les processus série-parallèle. Dans le cadre des posets série-parallèle, cette formule est un corollaire immédiat de la formule récursive de Möhring [Mö89] suivante (adaptée à la concurrence).

Fait 9.1.1

Soit P un processus série-parallèle et ℓ_P le nombre de ses exécutions.
Si P se décompose en série $P = P_1.P_2$, alors $\ell_P = \ell_{P_1} \cdot \ell_{P_2}$.
Si P se décompose en parallèle $P = P_1 \parallel P_2$, alors $\ell_P = \binom{n_1+n_2}{n_1} \cdot \ell_{P_1} \cdot \ell_{P_2}$, où n_1 (resp. n_2) correspond à la taille de P_1 (resp. P_2).

On en déduit directement le corollaire suivant, sur un processus Fork/Join, Afin d'en simplifier l'expression, nous utilisons la représentation en arbre ternaire des processus $\mathcal{FJ}$, cf. par exemple la Figure 8.5. Néanmoins l'adaptation à tout processus série-parallèle est directe (voir notre article [27]).

Étant donné un processus $P \in \mathcal{FJ}$, vu via sa représentation en arbre ternaire. Pour chaque nœud ν de P, on définit P_ν comme le sous-arbre enraciné en ν (et contenant tous les descendants de ν dans P). Pour chaque nœud ternaire ν dont les enfants de gauche à droite sont notés ν_1, ν_2 et ν_3, nous avons trois sous-arbres P_{ν_1}, P_{ν_2} et P_{ν_3}. Par exemple, sur la Figure 8.5, pour le nœud h, le sous-arbre P_{h_1} est réduit à i et le sous-arbre P_{ν_3} contient k et l.

Théorème 9.1.2

Soit P un processus de $\mathcal{FJ}$. Son nombre d'exécutions ℓ_P vaut

$$\ell_P = \prod_{\nu \text{ nœud ternaire}} \frac{|P_{\nu_1} \cup P_{\nu_2}|!}{|P_{\nu_1}|! \cdot |P_{\nu_2}|!}.$$

Soit n la taille de P, en gardant en mémoire tous les valeurs des nombres factorielles de 1 à n, le nombre ℓ_P est calculé, au pire cas, en $\Theta(n)$ opérations arithmétiques et en un seul parcours de P.

Via une application de la formule sur notre exemple de

la Figure 8.5, nous obtenons

$$\frac{2!\ 6!\ 2!}{1!\ 1!\ 4!\ 2!\ 1!\ 1!} = 60.$$

Il y a donc 60 exécutions pour ce processus-exemple. Notons que si l'arbre de P ne contient pas de nœud ternaire alors le processus contient une seule exécution.

Cette formule peut s'adapter simplement dans le cas général de processus série-parallèle. La première approche, proposée dans l'article [27] consiste à faire correspondre au processus série-parallèle un processus Fork/Join dans lequel on autorise des actions (nœuds) silencieuses, c'est-à-dire qu'elles n'interviennent pas dans la taille du processus.

Une autre possibilité consiste à représenter un processus série-parallèle via un arbre (de la même manière que pour $\mathcal{FJ}$) mais dont les nœuds sont d'arité quelconque, et le dernier enfant encode le sous-processus commençant après le Join. Dès lors il suffit de faire le produit sur l'ensemble des nœuds non unaires, avec la particularité de l'enfant le plus à droite qui est l'analogue de l'enfant ν_3 dans le cas $\mathcal{FJ}$.

9.2 Processus parallèle avec choix non-déterministe

Revenons sur le cas des processus sans synchronisation, mais avec choix non-déterministe (cf. l'article [19]). Rappelons qu'un tel processus peut être décomposé selon ses choix globaux (cf. Définition 7.2.2). Une idée naturelle dès lors pour compter le nombre d'exécutions d'un tel processus P consiste à sommer, sur l'ensemble des choix globaux, la formule des équerres d'un processus parallèle :

$$\ell_P = \sum_{C \text{ choix global de } P} \frac{|C|!}{\prod_{S \text{ sous-arbre de } C} |C|}.$$

L'efficacité de l'algorithme associé à cette expression est directement reliée à la structure du processus : combien de choix globaux sont induits par le processus. Et le résultat quantitatif suivant est plutôt négatif dans ce sens.

Théorème 9.2.1

Le nombre moyen k de choix globaux d'un processus parallèle avec choix non-déterministe de taille n (uniformément par rapport à l'ensemble des processus de taille n), vérifie, asymptotiquement, $k \sim \alpha \cdot \beta^n$ avec les constantes qui valent, approximativement $\alpha \approx 1.4408\ldots$ et $\beta \approx 1.1106\ldots$. Par ailleurs, la taille moyenne d'un choix global dans un processus de taille n est équivalente à $\gamma \cdot n$ où $\gamma \approx 0.49636\ldots$.

Ainsi, afin d'obtenir une formule dont l'évaluation est efficace, il faut éviter l'expansion des choix globaux puisqu'ils sont en noble exponentiel et de taille non négligeable. Notons, que pour deux choix globaux d'un processus fixé partageant des sous-arbres communs, de nombreux calculs sont dupliqués dans la formule précédente. En économisant ces calculs dupliqués, via une autre approche, nous allons réussir à obtenir une formule efficace.

Une approche qui porte ses fruits consiste à s'appuyer sur la spécification du processus étiqueté croissant et à sa traduction via la méthode symbolique. Nous allons simplement illustrer la méthode avec notre exemple

$$a.([(b.(c \parallel d)) + e] \parallel [f.((g + (h.(i \parallel j)) + k) \parallel l)]).$$

représenté dans la Figure 7.8. Voila la spécification des étiquetages croissants du processus (nous avons annoté les atomes $\mathcal{Z}$ afin de gagner en lisibilité) :

$$\mathcal{Z}_a{}^{\square} \star \Big([\mathcal{Z}_b{}^{\square} \star (\mathcal{Z}_c \star \mathcal{Z}_d) + \mathcal{Z}_e] \star [\mathcal{Z}_f{}^{\square} \star ((\mathcal{Z}_g + \mathcal{Z}_h{}^{\square} \star (\mathcal{Z}_i \star \mathcal{Z}_j) + \mathcal{Z}_k) \star \mathcal{Z}_l)] \Big).$$

Pour un processus fixé, sa spécification correspondante est évidemment non récursive, et le nombre d'atomes $\mathcal{Z}$ correspond à la taille du processus. Cette spécification se traduit via la méthode symbolique en le polynôme $P(z)$ (série génératrice exponentielle) associé au processus P.

$$\begin{aligned} P(z) &= \int_{z_1=0}^{z} [\int_{z_2=0}^{z_1} z_2^2 \mathrm{d}z_2 + z_1] \cdot \\ &\qquad [\int_{z_2=0}^{z_1} (z_2 + \int_{z_3=0}^{z_2} z_3^2 \mathrm{d}z_3 + z_2) z_2 \mathrm{d}z_2] \mathrm{d}z_1 \\ &= \frac{z^9}{405} + \frac{13\ z^7}{315} + \frac{2\ z^5}{15}. \end{aligned}$$

Ce polynôme $P(z)$ correspond à la série exponentielle énumérant les choix globaux étiquetés croissants de P. Ainsi, il y a $9!/405 = 896$ exécutions de l'unique choix global de taille 9, 208 exécutions comptant 7 actions et 16 pour les deux choix globaux de taille 5. La transformée de Laplace combinatoire, suivie de l'évaluation $z = 1$ de $P(z)$ donne ainsi 1120 exécutions distinctes pour le processus P.

Théorème 9.2.2

Soit P un processus parallèle avec choix non-déterministe de taille n. Le calcul du nombre d'exécutions est, au pire cas, réalisé en $\Theta(n^2)$ opérations arithmétiques, via l'approche par la méthode symbolique.

Démonstration (idées) : Dans le pire des cas, il y a n intégrations de polynômes de degré au plus n à effectuer. Notons que même en développant les produits de polynômes intermédiaires, la complexité reste identique. Par ailleurs, le choix non-déterministe est directement intégré en tant qu'addition dans le contexte des polynômes et donc ne modifie pas la complexité (au pire) du calcul si on l'effectue pour un processus purement parallèle. ■

Remarquons que la correction de l'algorithme repose sur les opérations sur les polynômes. En effet, dans le contexte des processus parallèle avec choix non-déterministes, la mise en série n'est valide que pour un atome suivi d'un sous-processus (et donc l'intégration du polynôme du

sous-processus est l'opération correcte à effectuée). Le nombre d'exécutions de deux sous-processus en parallèle fait intervenir un coefficient binomial (cf Fait 9.1.1) qui apparait naturellement dans le produit de deux polynômes (séries exponentielles). Enfin, le choix non-déterministe est géré par l'addition des coefficients des monômes de degré identiques associés aux sous-processus.

9.3 Processus série-parallèle avec choix non-déterministe

L'approche précédente, présentant une approche efficace pour le comptage des exécutions dans un processus parallèle avec choix non-déterministe est évidemment valide pour un processus purement parallèle, bien qu'elle soit moins efficace que celle basée sur la formule des équerres. Mais cette approche peut aussi être adaptée aux processus série-parallèle avec choix non-déterministe. Commençons par illustrer l'approche sur le processus Fork/Join de la Figure 8.5. L'idée consiste à descendre le plus profondément dans les compositions en série et en parallèle, plutôt que de descendre aux feuilles du processus. Puis, on remonte en effectuant les intégrations classiques (à la Greene) pour les deux opérateurs ${}^{\square}\star$ et $\star\,{}^{\blacksquare}$. Pour l'opérateur $\boxtimes$, on utilise aussi la traduction via la méthode symbolique et on associe le polynôme obtenu en faisant la convolution combinatoire (cf. Proposition 8.3.5) des deux polynômes associés aux sous-processus.

Rappelons la spécification des étiquetages croissants du processus exemple :

$$\begin{aligned}&\left(\mathcal{Z}_a{}^{\square}\star[(\mathcal{Z}_b{}^{\square}\star(\mathcal{Z}_d\star\mathcal{Z}_e)\star\mathcal{Z}_g{}^{\blacksquare})\star(\mathcal{Z}_c{}^{\square}\star\mathcal{Z}_f)]\right)\\&\boxtimes\left(\mathcal{Z}_h{}^{\square}\star[\mathcal{Z}_i\star\mathcal{Z}_j]\right)\boxtimes\left(\mathcal{Z}_k{}^{\blacksquare}\star\mathcal{Z}_l\right).\end{aligned}$$

On obtient dès lors le calcul suivant.

$$\begin{aligned}P(z)&=\left(\int_{z_1=0}^{z}(\int_{z_2=0}^{z_1}\int_{z_3=0}^{z_2}z_3^2\mathrm{d}z_3\mathrm{d}z_2\mathrm{d}z_1)\cdot(\int_{z_2=0}^{z_1}z_2\mathrm{d}z_2)\right)\\&\qquad*\left(\int_{z_1=0}^{z}z_1^2\mathrm{d}z_1\right)*\left(\int_{z_1=0}^{z}z_1\mathrm{d}z_1\right)\\&=\frac{z^7}{168}*\frac{z^3}{3}*\frac{z^2}{2}=\frac{60\;z^{12}}{12!}.\end{aligned}$$

La correction de cette approche du comptage des exécutions résulte de sa correction pour les processus parallèle, pour lesquels on ajoute la correction du calcul des opérations sur les séries exponentielles pour les produits $\star\,{}^{\blacksquare}$ et $\boxtimes$. Il est désormais clair que l'ajout du choix non-déterministe est direct concernant aussi bien la correction que l'efficacité de l'algorithme. Ainsi nous avons directement l'adaptation suivante.

Théorème 9.3.1

Soit P un processus série-parallèle avec choix non-déterministe de taille n. Le calcul du nombre d'exécutions de P est réalisé en $\Theta(n^2)$ opérations arithmétiques dans le pire des cas, via l'approche par la méthode symbolique.

Sur la Figure 9.1, on présente un exemple simple. Les trois sous-processus pour lesquels il y a un choix non-déterministes sont précédés par le nœud $\underline{+}$ et suivis par $\overline{+}$. Par apport au processus présenté sur la Figure 8.5,

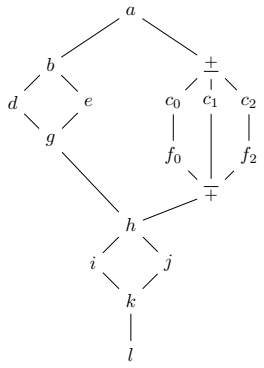


FIGURE 9.1 – *Un processus Fork/Join avec choix non-déterministe.*

on se rend facilement compte que ce nouveau processus avec choix non-déterministe contient deux fois plus d'exécutions contenant 12 atomes, et qu'il y a également des exécutions contenant 11 atomes. En calculant le polynôme associé, on trouve

$$P(z)=\frac{120\;z^{12}}{12!}+\frac{20\;z^{11}}{11!}.$$

Chapitre 10

Génération aléatoire d'exécutions

Lorsque l'on appréhende des systèmes concurrents, on se rend aisément compte qu'en raison de l'explosion combinatoire, effectuer de la vérification exhaustive sur tout l'espace d'états (i.e. toutes les exécutions possibles) n'est pas envisageable (cf. l'article de Sen [Sen07] et en particulier son introduction). Une possibilité pour contourner le problème consiste à réduire l'espace d'états, en exhibant, par exemple des suffixes d'exécutions identiques, et donc en donnant une relation d'équivalence sur des parties d'exécutions. Nous reviendrons sur cette approche dans les perspectives de nos travaux : cf. Section 13.2.

Une possibilité classique pour contourner le problème est basée sur du test aléatoire, la plupart du temps sans une véritable maîtrise de l'aléa de la génération sous-jacente pour construire les exécutions à vérifier. Cette technique passe à l'échelle sur les processus de grande taille. Néanmoins elle a un inconvénient fondamental : ne maîtrisant pas la distribution de probabilité à la base de la génération aléatoire, nous ne sommes pas en mesure de donner une mesure de couverture du test. En particulier, il a de grande chance que nous ne testions au final qu'un sous-ensemble très biaisé de l'espace d'états. L'article de Denise et al. [DGG+12] explicite ce problème en détail.

Nous voulons proposer une approche afin de générer des exécutions en maîtrisant la distribution de probabilité de génération. La manière la plus naturelle consiste à être en mesure d'effectuer de la génération uniforme, c'est-à-dire que chaque exécution a la même probabilité d'être sélectionnée (que les autres exécutions). Grâce à la distribution uniforme, l'usage d'outils statistiques nous permet de calculer en particulier l'espérance de la taille de l'espace couvert après un certain nombre de tirages aléatoires. Bien entendu, même si on n'est pas en mesure de générer uniformément, la connaissance de la distribution biaisée permet toujours d'obtenir des statistiques sur l'espérance de la couverture après un certain nombres de tirages aléatoires.

Notons toutefois, que la distribution uniforme a une propriété fort intéressante, révélée via le principe du collectionneur de coupons (cf. par exemple l'article [FGT92]). Lorsque l'on génère des objets via la distribution uniforme dans un ensemble de ℓ objets, alors le nombre de tirages nécessaires, en moyenne, afin de couvrir l'ensemble des objets (i.e. les avoir tous tirés une fois au moins) vaut $\ell \ln \ell$. Et de plus, c'est la génération via la distribution uniforme qui minimise l'espérance de la couverture.

10.1 Génération ad hoc

Afin de générer uniformément une exécution d'un processus donné, nous allons associer à chaque nœud de la structure un poids. Ce poids est fondamentalement lié aux formules des équerres. Puis un parcours du processus, en choisissant au fur et à mesure le prochain nœud à visiter en fonction de son poids, engendre une exécution aléatoire. Techniquement, le parcours est basé sur un tirage aléatoire dans un multi-ensemble. La preuve de validité (la génération est uniforme) est complètement reliée au choix du poids des nœuds. Une première version de cette approche a été présentée dans le contexte des processus parallèle dans les articles [19] et [8]. Nous présentons dans ce mémoire comment adapter l'approche aux processus $\mathcal{FJ}$.

Définition 10.1.1

Soit P un processus parallèle, et T son arbre syntaxique. Soit ν un nœud de T. L'action associée ν est accessible *si toutes ses actions ancêtres dans T ont été exécutées mais elle n'a pas été exécutée.*

Définition 10.1.2

Soit P un processus parallèle, et T son arbre syntaxique. Soit ν un nœud de T. Le poids de ν est la taille du sous-arbre enraciné en ν.

Algorithme 10.1.3

Soit P un processus parallèle. Supposons p le préfixe (d'exécutions) déjà parcouru. Soit $\mathcal{M}$ le multi-ensemble des actions accessibles de P (sachant les actions de p déjà exécutées), chacune associée à son poids. Pour choisir l'action suivante de l'exécution, on tire un élément de $\mathcal{M}$ suivant la distribution des poids (des éléments de $\mathcal{M}$).

Remarques :

- Au départ, le préfixe est vide et seule la racine du processus est accessible.
- L'algorithme s'arrête lorsque toutes les actions ont été exécutées.

— À chaque étape le multi-ensemble $\mathcal{M}$ est mis à jour, en enlevant l'action qui a été exécutée et en ajoutant les actions qui sont désormais accessibles.

Théorème 10.1.4

Soit P un processus parallèle de taille n. L'exécution construite par l'algorithme de génération est choisie uniformément parmi l'ensemble des toutes les exécutions du processus.

En implémentant l'évolution du multi-ensemble via un arbre équilibré dynamique, la complexité de génération est, au pire cas, en $\Theta(n \ln n)$ opérations sur l'arbre équilibré dynamique.

Notons qu'en modifiant la valeur des poids de chaque action du processus, on est en mesure de modifier la distribution de probabilités du tirage des exécutions. En particulier, en associant un poids 1 à chaque action, on définit une sorte d'uniformité locale : à chaque instant, on choisit la prochaine action à exécuter de manière uniforme (parmi les actions accessibles). Cette approche est appelée *isotrope* dans l'article [DGG+12]. Il est évident que le tirage global est biaisé : cette génération consiste à descendre dans l'arbre sémantique en choisissant uniformément une branche, sans tenir compte du sous-arbre enraciné à l'extrémité de cette branche.

Une caractéristique de l'Algorithme 10.1.3 est qu'il peut s'adapter à des familles de processus plus riches. Intéressons-nous désormais à la familles des processus Fork/Join. On prouve que pour adapter l'algorithme de génération uniforme à cette famille de processus, il est nécessaire d'utiliser une fonction de poids qui évolue après chaque exécution d'une action (pour les actions non choisies, donc restant accessibles).

Définition 10.1.5

Soit P un processus $\mathcal{FJ}$, et T son DAG syntaxique. Une action de P est dite accessible *si tous ses ancêtres dans T ont été exécutés mais elle n'a pas été exécutée.*

Soit P un processus $\mathcal{FJ}$, et T son DAG syntaxique. Soit p le préfixe d'une exécution et $\tilde{T}$ le DAG construit à partir de T en élaguant chaque action de p. On remarque que les nœuds source dans $\tilde{T}$ sont les (seules) actions accessibles. Soit ν un nœud accessible de $\tilde{T}$. On note $\mathcal{D}_\nu$ le DAG composé de ν et de ses descendants.
Soit μ un nœud accessible de $\tilde{T}$, différent de ν (s'il existe). On note j^μ_ν, la racine du sous-DAG $\mathcal{D}_\nu \cap \mathcal{D}_\mu$. Nous remarquons que le nœud j^μ_ν est, dans le DAG T, le nœud Join associé au nœud Fork qui met en parallèle les deux sous-processus, l'un contenant ν et l'autre contenant μ.
On note $\mathcal{J}_\nu$ l'ensemble des nœuds j^μ_ν, lorsque μ décrit l'ensemble des nœuds accessibles de $\tilde{T}$ (distincts de ν). Les nœuds ancêtres d'un nœud $j \in \mathcal{J}_\nu$, dans $\tilde{T}$ sont partitionnés en deux composantes connexes : on note $\mathcal{A}^\mu_\nu$ celle contenant ν et $\mathcal{A}^\mu_\nu$ l'autre composante.

Définition 10.1.6

Soit T un DAG, p le préfixe d'une exécution et $\tilde{T}$ le DAG issu de T en élaguant les actions de p. Soit ν un nœud accessible de $\tilde{T}$. Le poids de ν est

$$\mathrm{pds}(\nu) = \prod_{\mu \in \mathcal{J}_\nu} \frac{|\mathcal{A}^\mu_\nu|}{|\mathcal{A}^\mu_\nu| + |\mathcal{A}^\mu_\nu|}.$$

Remarques :
— La distribution des poids des nœuds accessibles est une distribution de probabilités.
— Le poids du même nœud accessible peut évoluer lorsque le préfixe de l'exécution croît.

Théorème 10.1.7

L'Algorithme 10.1.3 s'applique aux processus $\mathcal{FJ}$ avec la notion de poids définie en Définition 10.1.6.
La complexité temporelle est en $O(n^2)$ opérations arithmétiques.

Démonstration (idées) : La correction de l'algorithme réside dans le fait que la définition de poids des nœuds accessibles est une distribution de probabilités, directement reliée à la formule des équerres (cf. Théorème 9.1.2). La complexité de l'algorithme est due à la mise à jour à chaque étape de la distribution des poids des actions accessibles. ■

Nous remarquons que les deux algorithmes présentés dans notre article [27] afin de générer des exécutions sont distincts de l'algorithme précédent. Leur complexité au pire est du même ordre de grandeur, mais la mesure de coût moins onéreuse que le coût d'une opération arithmétique (le coût unitaire correspond à une écriture mémoire). En effet, ces autres versions, pour l'une basée sur le décomposition suivante récursive (*bottom-up*) d'un processus. Voilà sa description :
— si le processus est une séquence d'atome, il n'y a qu'une exécution possible
— si la racine du processus est un nœud Fork, on appelle récursivement l'algorithme sur chacun des deux sous-processus en parallèle ; puis on entrelace les deux exécutions obtenue ; enfin le résultat de l'entrelacement est précédé par le nœud Fork et suivi par une exécutions du sous-processus final (du dessous).
Le second algorithme est le dual du précédent, dans le sens qu'il s'agit de l'approche *top-down* correspondante à la version *bottom-up*. En implantant ces diverses versions pour la génération aléatoire uniforme d'exécutions, nous nous rendons compte que l'algorithme du Théorème 10.1.7 est moins efficace. Le calcul des poids est la tâche prépondérante dans la construction de l'exécution aléatoire. Dans le contexte des processus parallèle, le poids des nœuds accessibles n'évoluent pas au fil de l'exécution de l'algorithme. En réalité, la somme des poids du multi-ensemble est décrémenté de 1 à chaque étape (car les processus sont simplement parallèle). Dans le cadre des processus Fork/Join, même si on remplaçait les poids

(qui sont des fractions) par leur numérateur multiplié par le plus petit multiple commun de l'ensemble des dénominateurs des fractions, à chaque étape le poids d'un nœud évoluerait tout de même, et nous n'arriverions donc pas à une aussi bonne efficacité que celle exhibée dans le cadre des processus parallèle.

Enfin remarquons que l'approche présentée dans le cas des processus $\mathcal{FJ}$ pourrait aussi être adaptés aux processus avec choix non-déterministe. Néanmoins, afin de calculer à chaque étape la distribution de probabilités, des calculs de polynômes (similaires à ceux présentés dans la Section 9.2) semblent nécessaires et par conséquent, la complexité temporelle est encore davantage impactée.

10.2 Génération récursive

Nous avons remarqué dans la Section 9.2 que la construction de la série génératrice exponentielle associée à un processus nous permet de compter efficacement le nombre d'exécutions de ce dernier. Dans cette partie, nous allons expliquer comment, en ajoutant un peu d'information à cette série (qui est en réalité un polynôme), nous sommes en mesure d'utiliser les contextes de la génération récursive, historiquement présentée dans le livre [NW75] et, en particulier, adaptée au contexte de la méthode symbolique dans l'article de Flajolet et al. [FZC94]. Dans son manuscrit de thèse, Molinero [Mol05] adapte la méthode récursive pour les étiquetages croissants et la génération exhaustive. Néanmoins, son approche n'est pas tout à fait suffisante pour notre contexte. Nous ne voulons pas construire une structure croissante. Nous avons une structure non étiquetée fixée, et nous souhaitons l'étiqueter de façon croissante.

La démarche que nous présentons a été développée dans l'article [19] Nous allons à nouveau illustrer notre propos à l'aide de l'exemple suivant :

$$a.([(b.(c \parallel d)) + e] \parallel [f.((g + (h.(i \parallel j)) + k) \parallel l)]).$$

La spécification croissante associée est la suivante (après avoir retiré les annotations des atomes $\mathcal{Z}$).

$$\mathcal{Z}^{\square} \star \Big([\mathcal{Z}^{\square} \star (\mathcal{Z} \star \mathcal{Z}) + \mathcal{Z}] \star [\mathcal{Z}^{\square} \star ((\mathcal{Z} + \mathcal{Z}^{\square} \star (\mathcal{Z} \star \mathcal{Z}) + \mathcal{Z}) \star \mathcal{Z})]\Big).$$

Le premier obstacle à surmonter afin de pouvoir générer une exécution consiste à être capable d'identifier le choix global associé. Nous allons donc utiliser une nouvelle marque $\mathcal{Y}$ que nous annotons afin de distinguer les différents sous-processus d'un choix non-déterministe. On obtient donc

$$\mathcal{Z}^{\square} \star \Big([\mathcal{Y}_b \cdot \mathcal{Z}^{\square} \star (\mathcal{Z} \star \mathcal{Z}) + \mathcal{Y}_e \cdot \mathcal{Z}] \star [\mathcal{Z}^{\square} \star ((\mathcal{Y}_g \cdot \mathcal{Z} + \mathcal{Y}_h \cdot \mathcal{Z}^{\square} \star (\mathcal{Z} \star \mathcal{Z}) + \mathcal{Y}_k \cdot \mathcal{Z}) \star \mathcal{Z})]\Big).$$

Lors du calcul de la série génératrice, il faut la voir comme un polynôme en z, et ne pas développer les expressions contenant les variables y (ce qui impliquerait un nombre exponentiel de monômes, en moyenne – cf. Théorème 9.2.1).

$$\begin{aligned} P(z) &= \int_{z_1=0}^{z} [y_b \int_{z_2=0}^{z_1} z_2^2 \mathrm{d}z_2 + y_e z_1] \cdot \\ &\qquad [\int_{z_2=0}^{z_1} (y_g z_2 + y_h \int_{z_3=0}^{z_2} z_3^2 \mathrm{d}z_3 + y_k z_2) z_2 \mathrm{d}z_2] \mathrm{d}z_1 \\ &= \int_{z_1=0}^{z} [\frac{y_b z_1^3}{3} + y_e z_1] \cdot [\frac{(y_g + y_k) z_1^3}{3} + \frac{y_h z_1^5}{15}] \mathrm{d}z_1 \\ &= \frac{y_b y_h z^9}{405} + \left(\frac{y_b(y_g + y_k)}{9} + \frac{y_e y_h}{15}\right) \frac{z^7}{7} + \frac{y_e(y_g + y_k) z^5}{15}. \end{aligned}$$

Plusieurs possibilités s'offrent désormais. Soit on souhaite générer un choix global de taille quelconque, selon la distribution uniforme sur les exécutions (de toute taille). On effectue alors de la génération récursive sur le polynôme en z ce qui exhibera pour chacun des choix non-déterministe, une variable y annotée avec le sous-processus choisi. Une fois le choix global obtenu, on peut utiliser le générateur uniforme d'exécutions dans le processus parallèle basé sur ce choix.

Si l'on préfère effectuer une génération seulement parmi les choix globaux d'une taille donnée, il suffit d'isoler le monôme adéquat (en z) puis d'effectuer l'approche précédente uniquement sur ce monôme.

Théorème 10.2.1

Soit un processus parallèle avec choix non-déterministe de taille n. L'algorithme de génération aléatoire uniforme d'exécutions se comporte en $O(n^2)$ opérations arithmétiques et en $O(n^2)$ pour la complexité spatiale.

Démonstration (idées) : La complexité de l'algorithme réside dans la construction de la série génératrice (le polynôme) puis dans la génération récursive afin d'isoler un choix global. Son étiquetage est plus efficace que ces deux premières étapes. ■

10.3 Génération de préfixes d'exécutions

Dans un programme contenant au minimum quelques dizaines d'actions, il ne semble pas raisonnable de viser une couverture de l'ensemble de toutes les exécutions (même une couverture partielle, non négligeable face à l'espace total d'états). Par conséquent, une possibilité consiste à couvrir intégralement des préfixes d'exécutions, d'une certaine longueur, et d'effectuer du test sur ces préfixes.

Une fois la longueur t des préfixes choisie, l'on souhaite générer les préfixes (de longueur t) des exécutions de manière uniforme, parmi l'ensemble des préfixes possibles. En effet, c'est toujours cette distribution pour laquelle les statistiques pour la couverture des préfixes de longueur t sont les meilleures. Cette distribution uniforme sur les préfixes n'est pas obtenue en utilisant la distribution uniforme sur les exécutions (complètes) puis en s'arrêtant une fois un préfixe de longueur t généré.

Voilà l'approche que nous allons utiliser. Nous l'avons introduite dans l'article [23] pour deux applications. La première consiste, comme dans le contexte des formules des équerres, au comptage du nombre de préfixes d'exécutions de longueur fixée pour un processus donné. La deuxième application consiste en la génération aléatoire uniforme de préfixes de longueur donnée. L'idée est la suivante : chaque préfixe d'une exécution doit pouvoir être considéré comme une nouvelle exécution complète. Autrement dit, il faut être en mesure d'encoder le fait de pouvoir définir tout étiquetage croissant d'une partie des actions comme étant une exécution. Pour ce faire, on va introduire une nouvelle action ϵ muette, associée à chaque action du processus initial : en quelque sorte, elle élague les descendants de son action associée ; elle ne doit pas être étiquetée ; et enfin son poids doit être nul. Dans le contexte de la combinatoire analytique [FS09], c'est l'objet neutre noté ϵ. Ainsi, pour chaque action α suivie par un sous-processus (notée $\alpha.Q$), on complète en le processus $\alpha.(Q+\epsilon)$. De plus pour les processus du type, une action α suivie par la mise en parallèle de plusieurs sous-processus : $\alpha.(Q_1 \parallel \cdots \parallel Q_r)$ est modifié en $\alpha.((Q_1+\epsilon) \parallel \cdots \parallel (Q_r+\epsilon))$. Notons qu'on pourrait avoir dans le processus final deux nœuds + consécutifs (qui d'un point de vue combinatoire analytique et analyse quantitative n'est pas un problème).

Nous appliquons la démarche précédente sur l'exemple de processus de la Figure 7.8. Sur la Figure 10.1, nous avons complété (en rouge) le processus avec les atomes muets ϵ. La spécification s'adapte aisément.

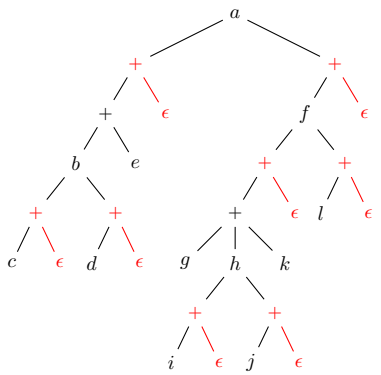


FIGURE 10.1 – *Processus complété pour obtenir une distribution uniforme sur les préfixes d'exécutions de même longueur*

$$\mathcal{Z}_a{}^{\square} \star \Big(([\mathcal{Z}_b{}^{\square} \star ((\mathcal{Z}_c{+}\epsilon) \star (\mathcal{Z}_d{+}\epsilon)) + \mathcal{Z}_e]{+}\epsilon) \star ([\mathcal{Z}_f{}^{\square} \star \Big(((\mathcal{Z}_g + \mathcal{Z}_h{}^{\square} \star ((\mathcal{Z}_i{+}\epsilon) \star (\mathcal{Z}_j{+}\epsilon)) + \mathcal{Z}_k){+}\epsilon) \star (\mathcal{Z}_l{+}\epsilon) \Big)]{+}\epsilon) \Big).$$

En oubliant les annotations des actions, nous obtenons le polynôme suivant.

$$P(z) = \frac{z^9}{405} + \frac{z^8}{45} + \frac{41\,z^7}{315} + \frac{41\,z^6}{90} + \frac{16\,z^5}{15} + \frac{5\,z^4}{3} + \frac{5\,z^3}{3} + \frac{3\,z^2}{2} + z.$$

En extrayant le monôme de degré t de ce polynôme et en multipliant le coefficient par $t!$, on obtient le nombre de préfixes d'exécutions du processus de longueur t. Par exemple, il y a $\frac{16}{5} \cdot 5! = 384$ préfixes de longueur 5.

En ajoutant les annotations pour toutes les actions, via les variables formelles $\mathcal{Y}$, on est désormais capable, via la génération récursive d'extraire uniformément t actions qui composent le préfixe, puis on peut générer un étiquetage croissant, par exemple via notre algorithme 10.1.3.

Théorème 10.3.1

Soit un processus parallèle avec choix non-déterministe de taille n. Un pré-calcul de complexité en $O(n^2)$ opérations arithmétiques permet par la suite, de générer uniformément des préfixes d'exécutions (de même taille) en $O(n^2)$ opérations arithmétiques.

Nous terminons ce chapitre avec un tableau comparatif de performances en utilisant deux types de distribution pour la génération de préfixes d'exécutions. Comme nous l'avons expliqué, la génération uniforme est la plus efficace si l'on a un souci de couverture. *Néanmoins, en utilisant une autre distribution, cette efficacité est-elle réellement impactée ?*

Afin d'effectuer une comparaison, nous avons choisi la distribution que nous qualifions d'isotrope. Après avoir généré un préfixe de longueur r, nous choisirons l'action suivante uniformément parmi l'ensemble des actions accessibles.

Lg. des préfixes	1	2	3	4	5
Nb. de préfixes	4	14	43	115	265
Espérance du tps. de couverture					
Isotrope	8	47	223	972	4343
Unif. globale	8	45	187	612	1646
Gain	0%	4.5%	19%	59%	164%

Lg. des préfixes	6	7	8
Nb. de préfixes	564	1201	2877
Espérance du tps. de couverture			
Isotrope	24087	137174	914313
Unif. globale	3891	8993	21719
Gain	519%	1425%	4110%

Les abréviations correspondent à : Lg. : Longueur ; Nb. : Nombre ; tps. : temps ; Unif. : Uniformité.

Les calculs ont été effectués sur un processus parallèle contenant 125 actions. On lit le tableau ainsi : parmi les 564 préfixes d'exécutions de taille 6, l'espérance du temps de couverture de l'ensemble des préfixes de taille 6 nécessite de générer aléatoirement 24087 préfixes en utilisant la distribution isotrope, alors que 3891 préfixes générés suffisent via la distribution uniforme globale.

Troisième partie

Détours combinatoires, perspectives et conclusion générale

Chapitre 11

Processus d'évolution

Les arbres croissants peuvent être construits et analysés grâce à un processus dynamique d'évolution. Les modèles classiques dans ce contexte sur usuellement appelés arbres récursifs. De nombreux articles se sont penchés sur les problèmes découlant de cette approche. Les premiers articles ont été écrits par Meir et Moon [Moo74, MM74] and [MM78]. On y trouve de nombreuses applications pour ces modèles.

L'aspect dynamique du processus repose sur le fait que des nœuds sont ajoutés au fur et à mesure à la structure en construction. À chaque fois qu'un nœud est greffé, son étiquette correspond au numéro d'arrivée du nœud. Dans l'article [22], nous avons étudié le paramètre *longueur de coupe* de différentes familles d'arbres croissants.

Dans la section suivante, nous définissons un nouveau modèle d'évolution, dans lequel plusieurs nœuds peuvent apparaître simultanément, et par conséquent portent la même étiquette.

11.1 Arbres croissants avec répétitions d'étiquettes

Nous allons définir un modèle combinatoire d'arbres croissants dans lesquels les étiquettes peuvent se répéter.

Définition 11.1.1

Un arbre binaire croissant avec répétitions *est composé d'une structure classique d'arbre binaire (enracinée, plane) associée à un étiquetage des nœuds internes tel que plusieurs nœuds peuvent avoir la même étiquette, et tout chemin de la racine vers une feuille contient une suite strictement croissante d'étiquettes.*

Dans notre contexte, nous supposons que tous les entiers de 1 jusqu'à l'étiquette la plus grande apparaissent au moins une fois dans l'arbre. Cette contrainte pourrait être relâchée dans un contexte combinatoire différent. Nous pourrions aussi, par exemple, relâcher la contrainte de croissance stricte sur un chemin de la racine vers les feuilles en une croissance non-stricte. D'un point de vue concurrence, ce relâchement n'a guère de signification.

Nous définissons la *taille* d'un arbre binaire croissant avec répétitions comme étant son nombre de feuilles (non étiquetées).

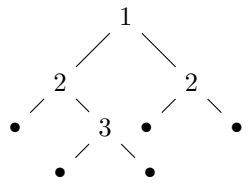


FIGURE 11.1 – *Arbre binaire croissant avec répétition (taille 5 ; nombre d'étiquettes distinctes 3)*

Nous remarquons spécifiquement sur la Figure 11.1 que, si nous souhaitons faire croître l'arbre, il faut placer au moins une étiquette 4 dans l'une des feuilles.

Approche récursive : Soit $B_{r,n}$ le nombre d'arbres binaires croissants avec répétitions contenant n feuilles (donc $n-1$ nœuds internes) et r étiquettes distinctes. Nous obtenons

$$\begin{cases} B_{1,2} = 1 \\ B_{1,n} = 0 \qquad \text{si } n \neq 2 \\ B_{r,n} = \displaystyle\sum_{\ell=1}^{n-r} \binom{n-\ell}{\ell} B_{r-1,n-\ell}. \end{cases}$$

En effet, pour construire un arbre à r étiquettes de taille n, à partir d'un arbre A à $r-1$ étiquettes et de taille $n-\ell$, il faut remplacer ℓ feuilles par ℓ nœuds (chacun parent de deux feuilles).

En regroupant les arbres par taille, nous obtenons l'énumération $B_n = \sum_r B_{r,n}$. Nous remarquons que cette suite est enregistrée dans OEIS, sous la référence A171792, mais sans explication combinatoire. Les premiers coefficients, pour $n \geqslant 2$ sont

$$1, 2, 7, 34, 214, 1652, 15121, 160110, 1925442, 25924260, \ldots$$

La suite de l'analyse peut être faite directement sur la suite récursive, ou alors nous pouvons tenter de transformer la récurrence en une équation fonctionnelle vérifiée par la série génératrice associée. Nous nous apercevons aussi que l'approche constructive directe via les séries génératrices est possible, mais dans le contexte des séries ordinaires mais pour des objets étiquetés. La spécification via des séries exponentielles ne nous semble pas naturelle car nos objets ne semblent pas décomposables.

Approche via les séries génératrices : Soit $B_r(z)$ la série génératrice ordinaire, dont la variable z marque

les feuilles et telle que le coefficient $[z^n]B_r(z)$ représente le nombre d'arbres de taille n avec r étiquettes. Pour construire la série $B_{r+1}(z)$, à partir de $B_r(z)$, chaque feuille des structures énumérées par $B_r(z)$, peut être ou non remplacée par une nouvelle paire de feuilles (implicitement, nous remplaçons une feuille par un nœud interne non compté par z et une paire de feuille z^2). Nous obtenons donc

$$B_{r+1}(z) = B_r(z+z^2) - B_r(z) \qquad B_1(z) = z^2.$$

Il faut faire attention au cas extrême : au moins une feuille doit être remplacée, donc nous retirons le cas où rien ne se passe : $B_r(z)$. En définissant la série $B(z) = \sum_{r=1}^{\infty} B_r(z)$, par téléscopage nous obtenons

$$B(z) = \frac{1}{2}\left(z^2 + B(z+z^2)\right).$$

Finalement, le nombre d'arbres binaires croissants avec répétitions de taille n est $[z^n]B(z)$, puisqu'il s'agit d'une série ordinaire.

Théorème 11.1.2

Soit B_n le nombre d'arbres binaires croissants avec répétitions et n feuilles. Asymptotiquement lorsque n tend vers l'infini, nous avons

$$B_n = \Theta\left(n^{\frac{-1}{\ln 2}}\ \ln(2)^{-n}\ (n-1)!\right).$$

Démonstration (méthode) : L'idée consiste a étudié directement la suite via des minorations et majorations. ■

La même approche peut être appliquée à de nombreux modèles arborescents. Ces modèles croissants avec répétitions sont d'un point de vue combinatoire suffisamment riches pour apporter une nouvelle vision concernant des modèles de DAG. Par ailleurs, sur cet exemple particulier, nous obtenons un comportement asymptotique de la suite énumérative non classique.

D'un point de vue combinatoire, de nombreuses questions sont soulevées par ce modèle. *Dans un grand arbre, en moyenne combien y-a-t-il d'apparitions de chaque valeur d'étiquette ? Typiquement, y-a-t-il beaucoup de nœuds internes ? Dans un grand arbre général croissant avec répétitions, lorsque la taille tend vers l'infini, à quoi ressemble la distribution du nombres d'arbres avec k nœuds internes (lorsque k varie) ?*

11.2 Arbres binaires compressés

Dans un contexte, qui semble à première vue éloigné, Flajolet et al. [FSS90] se sont intéressés à la compressions des arbres simplement générés. Plus précisément, d'un point de vue quantitatif, ils ont calculé le gain de taille de la compression de ces familles d'arbres. L'idée originelle est la suivante : étant donné un arbre, on ne souhaite représenter les sous-arbres identiques qu'une seule fois. Pour ce faire, un parcours de l'arbre est fixé, par exemple le parcours préfixe et à chaque fois qu'un un sous-arbre déjà rencontré réapparaît, plutôt que il est remplacé par un pointeur vers la première occurrence du sous-arbre. À la fin du processus de compression, on se retrouve avec un DAG (encodant l'arbre originel). Par exemple, la Figure 13.1 présente la forme compressée de l'arbre sémantique introduit dans la Figure 7.4. Flajolet et al. ont prouvé que, asymptotiquement, la taille du DAG est négligeable par rapport à la taille de l'arbre initial. Plus précisément, si la taille de l'arbre originel est n, alors, en moyenne, la taille du DAG (en nombre de nœuds) vaut $\Theta(n/\sqrt{\ln n})$.

Dans l'article [29], en cours de soumission, nous nous intéressons à la classe de DAG induite par la compression d'arbres binaires. Nous étudions ainsi, la structure pour elle-même, sans l'aspect algorithmique de la compression. En effet, l'approche de Flajolet et al. ne permet notamment pas de compter ces structures particulières de DAG.

Notons en particulier que via le parcours préfixe du DAG, nous obtenons un étiquetage décroissant canonique des nœuds de la structure. L'énumération directe ne semble pas accessible pour le moment. Notons que ce ne sont pas des objets décomposables. Néanmoins, nous sommes en mesure de les construire en étudiant un processus de (dé)croissance contraint, et en particulier nous pouvons représenter ces DAG via des arbres décroissants binaires avec répétitions d'étiquettes et quelques contraintes structurelles supplémentaires. En particulier, l'étiquette ℓ doit apparaître avant l'étiquette $\ell+1$ (pour le parcours préfixe). Formellement, les DAG sont représentés via des arbres binaires étiquetés. Les répétitions d'une étiquette représentent les pointeurs vers la première occurrence de l'étiquette (dans le parcours préfixe). Ainsi le DAG peut être représenté par un arbre étiqueté :

- l'arbre a une structure binaire telle que chaque nœud interne possède deux descendants
- les contraintes d'étiquetage sont :
  - de la racine vers les feuilles, la suite d'étiquettes est strictement décroissante
  - il est possible d'avoir des étiquettes répétées dans le DAG
  - l'étiquette $\ell+1$ apparaît après l'étiquette ℓ dans le parcours préfixe du DAG
  - si l'étiquette ℓ est répétée, seule la première occurrence (dans le parcours préfixe) peut éventuellement avoir des descendants
  - deux nœuds distincts étiquetés par i et par j ne peuvent pas avoir la même paire de descendants étiquetée (k,ℓ).

La dernière condition sur l'unicité de la paire des descendants assure que, dans la structure, deux sous-arbres identiques ne seront représentés qu'une seule fois. Sur la Figure 11.2, nous représentons un DAG (issu de la compression d'un arbre binaire) et notre représentation en arbre décroissant (avec contraintes). La notion de taille qui nous importe dans ce contexte est le nombre de nœuds du DAG, qui correspond donc au nombre d'étiquettes distinctes dans la représentation en arbre. Dès lors, la suite

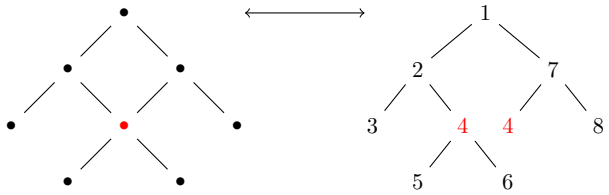


FIGURE 11.2 – *Un DAG et sa représentation sous forme d'arbre décroissant avec contraintes*

énumérative du nombre de structures commence ainsi

$$0, 1, 1, 3, 15, 111, 1119, 14487, 230943, 4395855, 97608831, \ldots$$

Via une structure combinatoire annexe, nous obtenons la récurrence suivante. Soit $n, p \in \mathbb{N}$,

$$\begin{cases} \gamma_{n+1,p} &= \sum_{i=0}^{n} \gamma_{i,p}\, \gamma_{n-i,p+i}, & \text{for } n \geqslant 1, \\ \gamma_{0,p} &= p+1, \\ \gamma_{1,p} &= p^2+p+1. \end{cases}$$

Notons que la suite $(\gamma_{n,0})_{n\in\mathbb{N}}$ énumère le nombre de DAG suivant leurs nombres d'étiquettes différentes (qui est la notion de taille adéquate au problème). L'étude du comportement asymptotique de cette suite semble très délicate. Pour le moment nous avons ajouté une contrainte afin d'être en mesure d'exhiber le comportement asymptotique d'une structure proche. Ainsi, nous fixons la profondeur droite des nœuds : la valeur maximale d'arêtes droites que nous traversons de la racine jusqu'à chaque nœud est fixée. En notant γ_n^k le nombre de DAG de taille n avec les profondeurs droites valant au plus k (indépendant de n), nous obtenons

$$\gamma_n^k = \Theta\left(n! \left(4\cos^2\left(\frac{\pi}{k+3}\right)\right)^n \cdot n^{-\frac{k}{2}-\frac{1}{k+3}-\left(\frac{1}{4}-\frac{1}{k+3}\right)\cos\left(\frac{\pi}{k+3}\right)^{-2}} \right).$$

Ces processus de croissance constituent un point de départ pour l'énumération de nombreux problèmes combinatoires dans le contexte de DAG.

Chapitre 12

Perspectives pour la logique quantitative

Dans l'ensemble des modèles présentés dans la partie dédiée à la logique quantitative, deux questions restent en suspens. La première est relative à l'effet Shannon. Nous avons été en mesure de prouver qu'une partie des probabilités est concentrée sur les fonctions de petite complexité. *Néanmoins, pouvons-nous améliorer l'approche proposée afin de prouver que, presque sûrement, toutes les fonctions sont de petite complexité ?* Bien que ce soit ce que nous conjecturons, nous n'avons pas d'idée actuellement pour réussir à prouver ce résultat : il semblerait que l'approche par approximations des expansions d'arbres minimaux ne soit pas suffisamment riche pour prouver le résultat global.

Par ailleurs, la question de la complexité moyenne d'une fonction, déjà soulevée par Gardy dans son article de synthèse [Gar06] de 2006 reste toujours sans réponse : *étant donné la distribution induite par les expressions arborescentes sur les fonctions booléennes, en choisissant aléatoirement une fonction, quelle est la complexité typique de cette moyenne ?*

En plus de ces questions ouvertes, d'autres modèles d'expressions booléennes pertinents aussi bien pour les questions résolues dans les chapitres précédents que pour ces deux dernières ouvertures nous intéressent fortement.

12.1 Arbres équilibrés

Rappelons que la distribution sur les fonctions booléennes induites par les arbres équilibrés s'est distinguée des autre car elle est concentrée sur un petit nombre de fonctions. Nos collègues [CGM15] et [BM15] ont démontré des résultats similaires sur d'autres modèles et ont donné une généralisation du résultat dans le cas des arbres dont toutes les feuilles sont, intuitivement, éloignées de la racine. Nous nous intéressons désormais à laisser tendre le nombre de variables considérées vers l'infini, en parallèle de la taille de l'arbre qui tend également vers l'infini.

Une autre question intéressante consiste à *contraindre l'étiquetage des nœuds internes des arbres.* En effet, nous pouvons représenter une formule formule 3-Sat sous une forme arborescente : tous les nœuds internes sont étiquetés avec le connecteur `et`, sauf ceux du dernier niveau qui sont étiquetés par des connecteurs `ou` ternaires. Une étape intermédiaire consiste à relâcher un peu cette contrainte sur l'étiquetage des nœuds internes, tout en conservant une contrainte relative à la profondeur des nœuds. Ces différentes distributions ne sont pas uniforme sur les expressions 3-Sat (qui ne sont pas toutes représentables), mais leur compréhension combinatoire et probabiliste permettrait d'apporter de nouveaux éclairages dans le contexte de la satisfaisabilité.

12.2 DAG booléens

Dans le contexte des fonctions booléennes calculées par des circuits booléens, (cf. par exemple [GGL95, Chapitre 40]), les sous-expressions communes sont partagées et par conséquent représentées une seule fois. La notion de taille utilisée usuellement correspond au nombre de sous-expressions (non réduites à une variable) distinctes. Ces expressions sont représentées naturellement par des graphes acycliques dirigés et, ainsi, cette notion de taille correspond au nombre de connecteurs dans le circuit.

Nous sommes, a peu près, en mesure d'énumérer les DAG de taille n issus de la compression d'arbres binaires. Plus précisément, nous avons une récurrence qui nous permet de les compter. Forts de ce résultat, nous pouvons nous intéresser à de telles structures décorées via des connecteurs ou variables de logique. nous pouvons donc étudier les structures classiquement appelées circuits booléens (ou DAG booléens) pour lesquels quelques résultats quantitatifs ont été exhibées, notamment par Shannon dès les années 1940 (cf. en particulier [RS42, Sha49]). Cependant, le type de questions qui nous ont intéressé dans la partie concernant la logique propositionnelle ne sont pour la grande majorité d'entre elles non résolues dans le contexte des circuits booléens.

Chapitre 13

Perspectives pour la concurrence quantitative

Nous l'avons observé dans le Chapitre 8 : certaines classes d'arbres croissants avec répétitions d'étiquettes sont susceptibles de représenter les exécutions de classes de processus concurrents. Comme nous l'avons explicité auparavant, un arbre croissant avec une seule répétition encode un processus parallèle avec une seule synchronisation. *Que se passe-t-il si nous nous intéressons à un modèle de croissance avec de nombreuses répétitions ?*

Cette piste, naturelle pour la concurrence, présente des modèles combinatoires qui n'ont, a priori, pas encore été explorés. Probablement, l'une des raisons principales, du point de vue de la combinatoire analytique en particulier, repose sur le fait que ces objets ne sont pas décomposables (récursivement) via les constructions classiques. Cependant, l'approche via le contexte des processus d'évolutions permet de construire, itérativement, des objets avec une croissance particulière (cf. en particulier les Sections 11.1 et 11.2).

Cette approche, très prometteuse, associe naturellement et de façon immédiate, la récurrence d'énumération des structures en parallèle de leur construction.

13.1 Arbres croissants avec répétitions contraints

D'un point de vue concurrence, les nœuds ayant la même étiquette peuvent être en correspondance avec une synchronisation. En s'autorisant des synchronisations sans (trop de) contraintes structurelles, nous construisons une nouvelle classe de DAG qui contient des structures qui ne sont plus série-parallèle.

La perspective à moyen terme consiste à comprendre l'ensemble des processus que nous pouvons exprimer, en contraignant plus ou moins le processus d'évolution. D'un point de vue concurrence, pour les modèles ayant du sens, il nous faudra comprendre la série bivariée marquant le nombre de nœuds et le nombre d'étiquettes distinctes, ce dernier paramètre ayant davantage de signification pour la concurrence.

Lorsque nous regardons un processus parallèle avec synchronisation via le modèle des arbres croissants avec répétitions, le problème fondamental repose sur le fait que tous les nœuds ayant la même étiquette peuvent avoir des descendants. Il s'agit par exemple du modèle simplifié étudié dans l'article [10] contenant une seule synchronisation. Dans ce mémoire (cf. Section 8.1), pour ce modèle très simplifié, nous avons d'ailleurs modifié, un peu les structures afin de n'autoriser qu'un seul des nœuds (avec la même étiquette) à avoir des descendants. Cela n'a pas entraîné davantage de difficultés techniques.

Dans le cadre des perspectives pour la concurrence, le premier but consiste à exhiber des contraintes supplémentaires au cœur des processus d'évolution afin de se rapprocher davantage de modèles significatifs pour la concurrence.

13.2 Compression d'arbres

Du fait de l'explosion en taille des arbres sémantiques, lorsque la taille des processus croît, plusieurs d'études se sont intéressées à diverses manières de représenter ces arbres de manière plus compacte. Par définition des arbres sémantiques, les préfixes communs des exécutions sont partagés. En particulier, dans son article [vG90], van Glabbeek définit la notion de classe d'équivalence de comportements. D'un point de vue combinatoire, une première manière de quantifier des équivalences de comportement consiste à partager les suffixes des exécutions. Nous remarquons directement ce partage consiste exactement en la compression de Flajolet et al. [FSS90] pour les arbres sémantiques des processus.

Sur la Figure 7.4, nous nous rendons compte, déjà sur ce petit exemple, que la compression pourrait être efficace. Sur les 30 exécutions, il n'y a que 12 suffixes de longueur 2 différents.

En considérant l'arbre sémantique, tel que nous le représentons (avec les étiquettes des actions en décoration), nous remarquons que la compression classique transforme l'arbre sémantique en un DAG sémantique avec partage de préfixes et de suffixes. Sur la Figure 13.1, nous avons remplacé les sous-arbres répétés par un pointeur vers leur première occurrence.

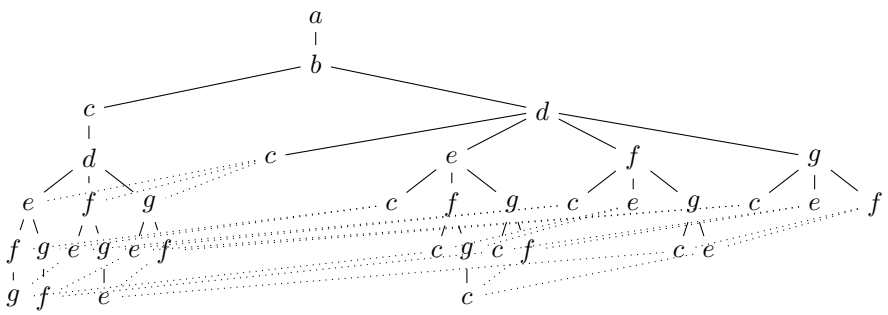


FIGURE 13.1 – *Le DAG sémantique issu de la compression de l'arbre sémantique de la Figure 7.4.*

Les arbres sémantiques ayant une structure très spéciale, en particulier n'étant pas des arbres simplement générés, l'approche proposée par Flajolet et al. ne permet pas d'analyser quantitativement, de façon directe, notre modèle.

Voilà le résultat clé nous permettant de calculer la taille moyenne des DAG sémantiques issus des processus généraux plans.

Proposition 13.2.1

Soit P un processus général parallèle et T son arbre syntaxique. L'ensemble des ensemble des feuilles des coupes admissibles de T est isomorphe à l'ensemble des nœuds du DAG sémantique de P.

Démonstration (idées) : L'idée fondamentale est la suivante. Soit P un processus de taille n. Soit deux préfixes d'exécutions distincts et de même longueur m, notés p_1 et p_2. Soit S_1 et S_2 les deux ensembles de suffixes, respectivement $\{s_{1,1}, s_{1,2}, \ldots, s_{1,r}\}$ et $\{s_{2,1}, s_{2,2}, \ldots, s_{2,r'}\}$ tels que pour tout $i \in \{1, \ldots, r\}$, $\langle p_1, s_{1,i}\rangle$ est une exécution et pour tout $i \in \{1, \ldots, r'\}$, $\langle p_2, s_{2,i}\rangle$ est une exécution.

Si les deux préfixes p_1 et p_2 contiennent le même ensemble d'actions (mais dans un ordre différent), alors les deux ensembles de suffixes S_1 et S_2 sont égaux. ■

Dans ces notions clé, nous nous apercevons que les coupes admissibles (cf. Définition 7.1.8) non étiquetées jouent un rôle central dans l'analyse des DAG sémantiques.

Théorème 13.2.2

Dans le contexte des processus parallèles généraux plans, la taille moyenne des DAG sémantiques issus des processus de taille n vaut

$$\bar{\mathcal{D}}_n \underset{n\to\infty}{\sim} \frac{2\sqrt{15}}{25}\, n \left(\frac{5}{4}\right)^{2n}.$$

Bien que la taille moyenne des DAG sémantiques croisse de manière exponentielle (avec pour facteur $\left(\frac{5}{4}\right)^2 \approx 1.5625\ldots$), cette taille moyenne est à mettre en relation avec la taille moyenne des arbres sémantiques dont la croissance est factorielle !

Concluons ces perspectives avec une résultat similaire pour les processus Fork/Join.

Théorème 13.2.3

Dans le contexte des processus $\mathcal{FJ}$, la taille moyenne des DAG sémantiques issus des processus de taille n vaut

$$\bar{\mathcal{D}}_n \underset{n\to\infty}{=} \Theta\left(n\, \beta^n\right),$$

avec $\beta \approx 1.1634\ldots$.

La croissance reste exponentielle, mais son facteur est relativement faible.

Chapitre 14

Conclusion

À travers ce mémoire, nous nous sommes intéressés à deux domaines de l'informatique, que sont la logique propositionnelle et la théorie de la concurrence. Notre but a consisté à relier la syntaxe des objets à leur sémantique via des études quantitatives et algorithmiques, en utilisant des outils de la combinatoire analytique mais aussi en en développant de nouveaux. Nous avons pu montrer qu'une compréhension fine de la combinatoire d'un modèle permet une compréhension globale de phénomènes complexes, en particulier dans le comportement des objets typiques. L'effet Shannon dans le cas des expressions logiques en est un exemple frappant, et c'est aussi la cas pour les formules des équerres concernant les structures croissantes en concurrence ou encore à la compression des arbres sémantiques qui chacun donne des algorithmes bien plus efficaces que les approches naïves. Ainsi, la compréhension en détail des structures permet d'optimiser les approches algorithmiques travaillant sur des structures complexes, en concentrant essentiellement les temps de calcul soit dans des pré-calculs, si possible, soit en éliminant dès que possible la complexité de calcul pour finalement retrouver un autre objet pour lequel la complexité de résolution du problème est nettement plus efficace. Nous pensons par exemple, à l'extraction préliminaire des choix non-déterministes qui permet par la suite d'utiliser les outils algorithmiques très efficaces pour les processus ne contenant que l'opérateur parallèle.

Pour la logique quantitative, mes perspectives à court terme se concentrent sur des études quantitatives fondées sur les modèles de DAG booléens (i.e. circuits booléens). Dans ce contexte, la complexité des fonctions diminue de façon naturelle car on ne compte qu'une seule fois les occurrences multiples dans leur syntaxe. Par conséquent, intuitivement, on a de bonnes raisons de penser que la distribution de probabilité sur les fonctions va beaucoup évoluer. Une fois que les questions autour de l'effet Shannon seront élucidées, les problématiques autour de la satisfaisabilité dans les circuits pourra devenir prépondérante.

Pour la concurrence quantitative, des problèmes importants se cristallisent autour du résultat de $\#P$-complétude du comptage des exécutions d'un processus quelconque. Je souhaite exhiber des sous-familles de processus, au-delà de la classe de processus série-parallèle pour lesquelles une approche de comptage des exécutions efficace est possible. Ces familles ne seront probablement pas décomposable au sens de la combinatoire analytique, et par conséquent, la génération aléatoire d'exécutions ne sera pas une application directe de la génération récursive. Il sera important dès lors de développer de nouveaux outils algorithmiques pour cette question fondamentale, aussi bien d'un point de vue théorique, que pour l'application des résultats à la concurrence ou à d'autres domaines.

A plus long terme, les briques algorithmiques que nous avons construites devront être assemblées au sein d'un logiciel permettant de faire de la vérification statistique de processus.

Bibliographie

[AB09] S. Arora and B. Barak. *Computational Complexity : A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[AM06] D. Achlioptas and C. Moore. Random k-SAT : Two moments suffice to cross a sharp threshold. *SIAM Journal of Computing*, 36(3) :740–762, 2006.

[AS12] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions : with Formulas, Graphs, and Mathematical Tables*. Dover Publications, 2012.

[BFS92] F. Bergeron, P. Flajolet, and B. Salvy. Varieties of increasing trees. In *CAAP '92, 17th Colloquium on Trees in Algebra and Programming, Rennes, France, February 26-28, 1992, Proceedings*, volume 581 of *LNCS*, pages 24–48. Springer, 1992.

[BK08] C. Baier and J.-P. Katoen. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.

[BLL98] F. Bergeron, G. Labelle, and P. Leroux. *Combinatorial Species and Tree-like Structures*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1998.

[BM15] N. Broutin and C. Mailler. And/or trees : a local limit point of view. Technical report, https ://arxiv.org/abs/1510.06691, 2015.

[Bop85] R. B. Boppana. Amplification of Probabilistic Boolean Formulas. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 20–29, 1985.

[BW91] G. Brightwell and P. Winkler. Counting linear extensions is #p-complete. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 175–181, 1991.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2) :244–263, 1986.

[CFGG04] B. Chauvin, P. Flajolet, D. Gardy, and B. Gittenberger. And/Or trees revisited. *Combinatorics, Probability and Computing*, 13(4-5) :475–497, July-September 2004.

[CGM15] B. Chauvin, D. Gardy, and C. Mailler. A sprouting tree model for random boolean functions. *Random Struct. Algorithms*, 47(4) :635–662, 2015.

[CGP99] E.M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT Press, 1999.

[CK98] A. Connes and D. Kreimer. Hopf algebras, renormalization and noncommutative geometry. *Comm. Math. Phys.*, 199(1) :203–242, 1998.

[DGG+12] A. Denise, M.-C. Gaudel, S.-D. Gouraud, R. Lassaigne, J. Oudinet, and S. Peyronnet. Coverage-biased random exploration of large models and application to testing. *STTT*, 14(1) :73–93, 2012.

[Die17] M. Dien. *Processus concurrents et combinatoire des structures croissantes : analyses quantitatives et algorithmes de génération aléatoire*. Thèse de doctorat, `https://www-apr.lip6.fr/~dien/these.pdf`, Université Pierre et Marie Curie, 2017.

[DNR04] R. David, K. Nour, and C. Raffalli. *Introduction à la logique : Théorie de la démonstration, 2ème édition*. Dunod, 2004.

[dPGGK16] É. de Panafieu, D. Gardy, B. Gittenberger, and M. Kuba. 2-xor revisited : Satisfiability and probabilities of functions. *Algorithmica*, 76(4) :1035–1076, 2016.

[DPRS12] A. Darrasse, K. Panagiotou, O. Roussel, and M. Soria. Biased Boltzmann samplers and generation of extended linear languages with shuffle. In *23rd International Meeting on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms, (AofA)*, pages 125–140, Montreal, Canada, June 2012.

[DR11] H. Daudé and V. Ravelomanana. Random 2-XORSAT phase transition. *Algorithmica*, 59(1) :48–65, 2011.

[Drm97] M. Drmota. Systems of functional equations. *Random Structures and Algorithms*, 10(1-2) :103–124, 1997.

[FGT92] P. Flajolet, D. Gardy, and L. Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics*, 39(3) :207–229, 1992.

[Fra86] J. Françon. Une approche quantitative de l'exclusion mutuelle. *ITA*, 20(3) :275–289, 1986.

[FS09] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[FSS90] P. Flajolet, P. Sipala, and J.-M. Steyaert. Analytic variations on the common subexpression problem. In *Automata, languages and programming (Coventry, 1990)*, volume 443 of *Lecture Notes in Comput. Sci.*, pages 220–234. Springer, New York, 1990.

[FZC94] P. Flajolet, P. Zimmermann, and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theor. Comput. Sci.*, 132(1) :1–35, September 1994.

[Gar06] D. Gardy. Random Boolean expressions. In *Colloquium on Computational Logic and Applications*, volume AF, pages 1–36. DMTCS Proceedings, 2006.

[GGL95] R. L. Graham, M. Grötschel, and L. Lovász, editors. *Handbook of Combinatorics (Vol. 2)*. MIT Press, Cambridge, MA, USA, 1995.

[God96] P. Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems : An Approach to the State-Explosion Problem*. Springer-Verlag New York, Inc., 1996.

[Gre83] D. H. Greene. *Labelled Formal Languages and Their Uses*. PhD thesis, Stanford University, Stanford, CA, USA, 1983. AAI8320712.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[Knu98] D. E. Knuth. *The art of computer programming, volume 3 : (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.

[Koz08] J. Kozik. Subcritical pattern languages for And/Or trees. In *Fifth Colloquium on Mathematics and Computer Science*, Blaubeuren, Germany, september 2008. DMTCS Proceedings.

[KP12] M. Kuba and A. Panholzer. Bilabelled increasing trees and hook-length formulae. *European Journal of Combinatorics*, 33(2) :248–258, 2012.

[Lal93] S. P. Lalley. Finite range random walk on free groups and homogeneous trees. *The Annals of Probability*, 21, 1993.

[LS97] H. Lefmann and P. Savický. Some typical properties of large And/Or Boolean formulas. *Random Structures and Algorithms*, 10 :337–351, 1997.

[Lup62] O. B. Lupanov. Complexity of formula realization of functions of logical algebra. *Problemy Kibernetiki, 3 :61-80, 1960. English translation : Problems of Cybernetics, Pergamon Press*, 3 :782–811, 1962.

[Lup65] O. B. Lupanov. On the realization of functions of logical algebra by formulae of finite classes (formulae of limited depth) in the basis $\wedge$, $\vee$, $\neg$. *Problemy Kibernetiki, 6 :5-14, 1961. English translation : Problems of Cybernetics, Pergamon Press*, 6 :1–14, 1965.

[Mil80] R. Milner. *A Calculus of Communicating Systems*. Springer Verlag, 1980.

[MM74] A. Meir and J. W. Moon. Cutting down recursive trees. *Mathematical Biosciences*, 21(3-4) :173–181, 1974.

[MM78] A. Meir and J. W. Moon. On the altitude of nodes in random trees. *Canad. J. Math.*, 30 :997–1015, 1978.

[Mol05] X. Molinero. *Ordered Generation of Classes of Combinatorial Structures*. Phd thesis, Universitat Politècnica de Catalunya, 2005.

[Moo74] J. W. Moon. The distance between nodes in recursive trees. In *London Math. Soc. Lecture Note Ser.*, volume 13, pages 125–132, 1974.

[MTZ00] M. Moczurad, J. Tyszkiewicz, and M. Zaionc. Statistical properties of simple types. *Mathematical Structures in Computer Science*, 10(5) :575–594, 2000.

[Mö89] R. H. Möhring. Computationally tractable classes of ordered sets. In I. Rival, editor, *NATO Advanced Study Institute on Algorithms and Order*, volume 255 of *Algorithms and Order*, pages 105–193. Springer-Verlag, 1989.

[NW75] A. Nijenhuis and H.S. Wilf. *Combinatorial algorithms*. Computer science and applied mathematics. Academic Press, New York, NY, 1975.

[Pei85] C. S. Peirce. On the algebra of logic : A contribution to the philosophy of notation. *American Journal of Mathematics*, 7 :180–202, 1885.

[Pet62] Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Universität Hamburg, 1962.

[PVW94] J. B. Paris, A. Vencovská, and G. M. Wilmers. A natural prior probability distribution derived from the propositional calculus. *Annals of Pure and Applied Logic*, 70 :243–285, 1994.

[Pó37] G. Pólya. Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen. *Acta Mathematica*, 68(1) :145–254, 1937.

[RS42] J. Riordan and C. E. Shannon. The number of two terminal series-parallel networks. *Journal of Math. Physics*, 21 :83–93, 1942.

[Sah89] N. Saheb. Concurrency measure in commutation monoids. *Discrete Applied Mathematics*, 24(1) :223–236, 1989.

[Sen07] K. Sen. Effective random testing of concurrent programs. In *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*, ASE '07, pages 323–332. ACM, 2007.

[Sha49] C. E. Shannon. The synthesis of two-terminal switching circuits. *Bell Systems Technical J.*, 28 :59–98, 1949.

[Sta86] R. P. Stanley. *Enumerative Combinatorics*. Wadsworth & Brooks/Cole, 1986.

[SU98] M. H. Sørensen and P. Urzyczyn. *Lectures on the Curry-Howard isomorphism*, volume 149. Studies in Logic and the Foundations of Mathematics, 1998.

[TD88] A. S. Troelstra and D. Van Dalen. *Constructivism in Mathematics : an Introduction*. Studies in Logic and the Foundations of Mathematics. North-Holland, 1988.

[Val84] L. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5 :363–366, 1984.

[Val89] A. Valmari. Stubborn sets for reduced state space generation. In *Advances in Petri Nets 1990 [10th International Conference on Applications and Theory of Petri Nets, Bonn, Germany, June 1989, Proceedings]*, pages 491–515, 1989.

[vD86] D. van Dalen. Intuitionistic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic : Volume III : Alternatives to Classical Logic*, pages 225–339. Reidel, Dordrecht, 1986.

[vG90] R. J. van Glabbeek. The linear time - branching time spectrum. In J. C. M. Baeten and J. W. Klop, editors, *CONCUR '90 Theories of Concurrency : Unification and Extension : Amsterdam, The Netherlands, August 27–30, 1990 Proceedings*, pages 278–297, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[Woo97] A. R. Woods. Coloring rules for finite trees, and probabilities of monadic second order sentences. *Random Structures and Algorithms*, 10(4) :453–485, 1997.