Systèmes de fonctions holonomes, application à la théorie des automates

Florent Koechlin

Cet article est le résumé de la thèse de doctorat [1] *de l'auteur ayant reçu l'accessit du prix de thèse Gilles Kahn 2022 décerné par la Société informatique de France.*

Ma thèse se divise en deux parties relativement indépendantes. Dans la première partie, j'ai étudié la pertinence des modèles d'expressions aléatoires utilisées pour l'analyse en moyenne des algorithmes ou pour faire des *benchmarks*. Dans la seconde partie, je me suis intéressé à la non-ambiguïté en théorie des automates, et notamment aux automates de Parikh non ambigus. Les deux parties se rejoignent dans la méthodologie utilisée pour les aborder, à savoir les séries génératrices et la combinatoire analytique.

Réductions d'arbres d'expressions en présence d'un élément absorbant

En informatique comme en mathématiques, il est courant de décrire des objets sous une forme condensée. Par exemple une expression régulière décrit un langage, une formule logique décrit une fonction booléenne, une formule de Presburger représente un ensemble de nombres, une expression arithmétique une fonction mathématique, etc.

Ces expressions se représentent naturellement sous la forme d'arbres, et sont ainsi l'entrée de nombreux algorithmes, dont on peut vouloir estimer l'efficacité. La mesure classique pour cela est la complexité au pire cas, mais il y a souvent une grande

1. https://www.theses.fr/2021UEFL2025.

DOI:10.48556/SIF.1024.21.173

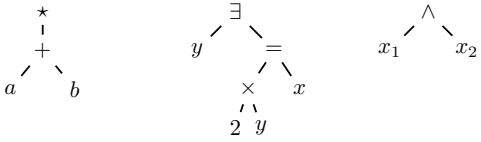


FIGURE 1. Exemple d'arbres représentant les expressions $(a + b)^*$, $\exists y,\ 2 \times y = x$, et $x_1 \wedge x_2$

différence entre ce qui est prédit au pire, et ce qui est observé à l'exécution. Une approche plus pratique consiste à tester les algorithmes sur des exemples de la vraie vie, mais avoir accès à de tels exemples, en nombre et de toutes les tailles, peut s'avérer difficile. L'analyse en moyenne est une autre approche classique plus pragmatique, qui est abordable dans certains cas mathématiquement, et qui peut se tester expérimentalement si on dispose d'un générateur aléatoire. Il faut cependant choisir une distribution sur les entrées de taille n, qui soit à même de simuler des exemples réels. Lorsqu'aucune information supplémentaire n'est connue sur la distribution des entrées, il est naturel de se tourner vers deux distributions standard :

— la distribution uniforme, pour laquelle chaque expression de taille n est équiprobable. Elle a l'avantage de donner la même probabilité à toute expression d'une taille fixée, et ainsi de bien couvrir toutes les possibilités. Elle est par exemple utilisée pour l'analyse en moyenne d'algorithmes de transformation d'expressions aléatoires en automate [12, 3] ;

— la distribution ABR pour les arbres binaires ou unaire-binaires. Le choix de cette distribution s'explique par la facilité avec laquelle on peut générer un arbre aléatoire de taille n, en temps linéaire : pour tirer un arbre de taille $n \geqslant 3$, il suffit de tirer un opérateur au hasard ; si l'opérateur est unaire, on génère récursivement son fils de taille $n-1$; si l'opérateur est binaire, on tire uniformément au hasard la taille k de son fils gauche, et on génère de façon récursive le fils gauche puis droit de taille k et $n-1-k$. Cette distribution, par rapport à la distribution uniforme, favorise les arbres plus équilibrés (cf. figure 2). Elle est notamment utilisée dans des générateurs de formules de logique temporelle linéaire (LTL) utilisées en vérification.

Redondances des arbres d'expression. On fait généralement la distinction entre la syntaxe d'une expression, c'est-à-dire l'arbre étiqueté, et sa sémantique, c'est-à-dire l'objet qu'il représente. Dans le code de génération des formules LTL aléatoires [14, 6], on peut remarquer qu'à aucun moment la sémantique des formules représentées n'est utilisée dans l'algorithme. Les arbres produits sont purement syntaxiques, ce qui peut paraître paradoxal dans la mesure où ils ont pour vocation à être considérés par les algorithmes qui vont les utiliser comme des formules logiques avec une sémantique précise. De même, si on regarde les articles utilisant

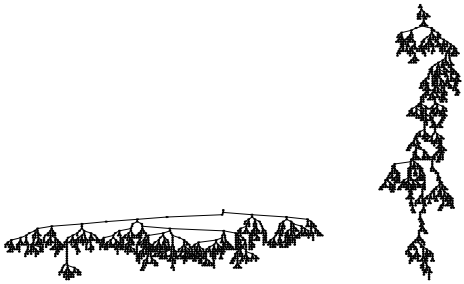


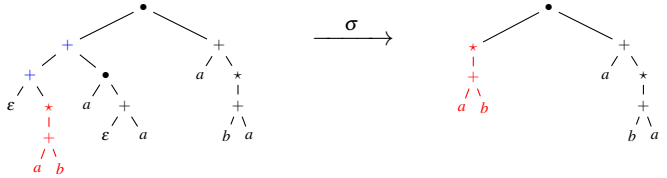
FIGURE 2. Un arbre ABR typique à gauche, de hauteur en moyenne $O(\log n)$, et un arbre typique uniforme à droite, de hauteur en moyenne $O(\sqrt{n})$.

la distribution uniforme pour l'analyse en moyenne d'algorithmes traitant des expressions régulières, les arbres aléatoires décrits pour l'analyse sont aussi purement syntaxiques.

Pour de nombreuses classes d'expressions syntaxiques, il existe pourtant des phénomènes de redondance : plusieurs expressions différentes représentent les mêmes objets (par exemple 0, $x+(-x)$, 0×42 et $\cos(\pi/2)$). Il apparaît ainsi immédiat que tirer au hasard une expression n'est pas du tout équivalent à tirer au hasard l'objet qu'elle représente ; certains objets sont sur-représentés par rapport à d'autres, selon le nombre d'expressions d'une taille donnée qui les décrivent. Dans ce cas, les analyses en moyennes et les *benchmarks* utilisant des expressions aléatoires sont-ils pertinents ?

Élément absorbant. Pour aborder cette question dans ma thèse, je me suis restreint à l'étude d'une redondance très simple, induite par la présence d'un élément absorbant. Un arbre $\mathscr{P}$ est dit absorbant pour un opérateur $\circledast$ si toute expression de la forme $\overset{\circledast}{\underset{\mathscr{P}\ T}{\wedge}}$ ou $\overset{\circledast}{\underset{T\ \mathscr{P}}{\wedge}}$, avec T une expression quelconque, est équivalente sémantiquement à $\mathscr{P}$. Par exemple, $(a+b)^*$ est absorbant pour l'union $+$ pour les expressions régulières sur les lettres a et b ; $\top$ est absorbant pour l'opérateur $\vee$; 0 est absorbant pour la multiplication $\times$, etc.

J'ai considéré une réduction particulière, notée σ. Étant donné un arbre fixé $\mathscr{P}$, appelé élément absorbant, et $\circledast$ un opérateur d'arité $a \geqslant 2$, appelé opérateur absorbant, la réduction σ d'une expression T est l'expression $\sigma(T)$ obtenue à partir de T en remplaçant de bas en haut par $\mathscr{P}$ tout sous-arbre de T dont la racine est $\circledast$ et l'un des fils est $\mathscr{P}$. Par exemple, dans le cas des expressions régulières, avec $\mathscr{P} = \overset{\star}{\underset{a\quad b}{+}}$ et $\circledast = +$, l'arbre suivant à gauche est simplifié par σ en l'arbre à droite (notez que $(b+a)^*$ n'est pas identifié comme étant égal à $\mathscr{P}$) :



Si les expressions admettent dans leur sémantique un élément absorbant, alors pour toute expression T, la réduction associée $\sigma(T)$ est une expression plus petite que T, qui lui est équivalente.

Contributions. J'ai étudié dans ma thèse le comportement en moyenne de la réduction σ pour des expressions suivant la distribution uniforme et la distribution ABR. Le fait d'étudier uniquement les conséquences de la présence d'un élément absorbant permet d'obtenir des propriétés générales sur de nombreuses classes d'expressions très diverses, et évite ainsi l'étude au cas par cas de la sémantique particulière de chacune des expressions étudiées. Pour cela, un outil de choix est la *combinatoire analytique* [8], qui s'est justement beaucoup appliquée à rassembler dans un même cadre unifié l'étude de classes combinatoires pourtant en apparence très différentes, à l'aide notamment de la méthode symbolique.

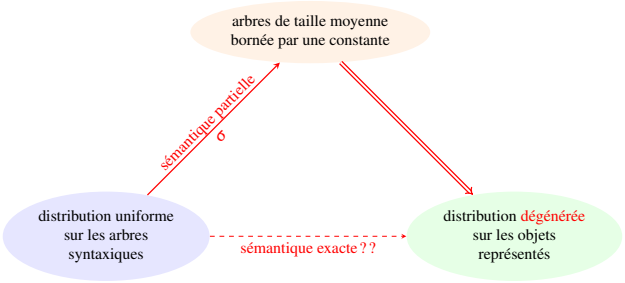


FIGURE 3. Résumé de la démarche : au lieu d'étudier directement la sémantique des expressions, on étudie à l'aide de la réduction σ un modèle intermédiaire, proche des arbres syntaxiques, mais qui suffit à démontrer la dégénérescence des expressions aléatoires uniformes.

Arbres d'expression uniformes. J'ai montré dans ma thèse que les redondances introduites par la seule présence d'un élément absorbant rendent la distribution uniforme dégénérée pour la plupart des expressions décrites par des systèmes d'équations. Cela inclut par exemple le cas des expressions régulières, décrites par l'équation unidimensionnelle :

$$\mathscr{L}_{\mathscr{R}} = \varepsilon + a + b + \overset{\star}{\underset{\mathscr{L}_{\mathscr{R}}}{|}} + \overset{\bullet}{\underset{\mathscr{L}_{\mathscr{R}}}{\wedge}}\underset{\mathscr{L}_{\mathscr{R}}}{} + \overset{+}{\underset{\mathscr{L}_{\mathscr{R}}}{\wedge}}\underset{\mathscr{L}_{\mathscr{R}}}{} . \tag{1}$$

ou encore les expressions régulières qui évitent deux étoiles consécutives, qui sont décrites par le système suivant :

$$\begin{cases} \mathscr{R} = \overset{\star}{\underset{\mathscr{S}}{|}} + \mathscr{S} \\ \mathscr{S} = a + b + \varepsilon + \overset{+}{\underset{\mathscr{R}}{\wedge}}\underset{\mathscr{R}}{} + \overset{\bullet}{\underset{\mathscr{R}}{\wedge}}\underset{\mathscr{R}}{} \end{cases} . \tag{2}$$

Plus précisément, j'ai démontré que sous certaines conditions naturelles, et en présence d'un élément absorbant, les systèmes de ce type décrivent des arbres d'expressions qui sont réduits considérablement par σ : en moyenne, la taille de la réduction $\sigma(T)$ d'un arbre de taille n est bornée par une constante indépendante de n. Ce phénomène reste vrai pour les moments d'ordre supérieur liés à la taille de $\sigma(T)$, qui sont aussi bornés par des constantes. Cela signifie que la distribution uniforme sur les arbres d'expressions n'est pas pertinente pour faire des *benchmarks*, ni pour faire une analyse en moyenne : avec cette distribution, tout problème polynomial sur des objets représentés par des expressions présentant un élément absorbant se résout en temps constant en moyenne.

En ajoutant quelques règles de réduction supplémentaires pour détecter une plus grande sous-famille d'expressions équivalentes à $(a+b)^*$, il est possible d'obtenir des résultats plus fins sur la classe des expressions régulières. En particulier, une expression régulière aléatoire uniforme de taille n formée avec deux lettres sur le modèle de l'équation (1) est en moyenne équivalente à une expression régulière de taille 77 lorsque n tend vers l'infini (cf. figure 4) ; de plus, asymptotiquement, entre 31 % et 46 % de ces expressions sont équivalentes à $(a+b)^*$, une expression de taille 4 seulement.

Arbres d'expressions ABR. Le comportement de la réduction σ est plus complexe lorsque les expressions sont des arbres unaire-binaires suivant la distribution ABR, qui est notamment utilisée pour générer des expressions LTL aléatoires. La taille moyenne après réduction dépend de la probabilité $p_{\circledast}$ d'apparition de l'opérateur absorbant. Intuitivement, il y a peu de réductions lorsque $p_{\circledast}$ est trop petit, et il y a beaucoup de simplifications lorsque $p_{\circledast}$ est proche de 1.

Plus précisément, si p_{I} est la probabilité de tirer un opérateur unaire, il existe deux seuils critiques, en $p_{\circledast} = 1/2$ et en $p_{\circledast} = (3-\pi)/4$, qui délimitent ainsi cinq régimes différents ; ces derniers couvrent toutes les possibilités, d'une réduction très faible en

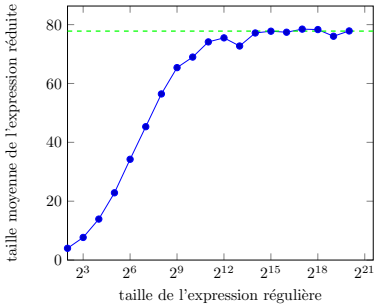


FIGURE 4. En appliquant quelques règles élémentaires de simplification, les expressions régulières aléatoires uniformes se réduisent considérablement.

$\Theta(n)$ à une réduction complète en $\Theta(1)$. Ces paliers ont été confirmés expérimentalement.

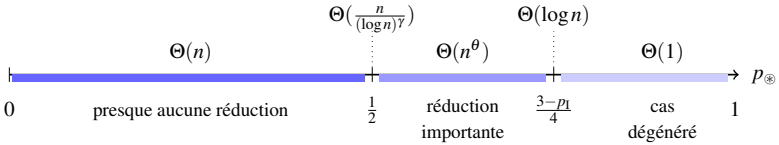


FIGURE 5. Les cinq régimes pour la taille moyenne de $\sigma(T)$, avec T un arbre ABR de taille n, selon la probabilité $p_\circledast$ de l'opérateur absorbant, avec p_I la probabilité des opérateurs unaires, $\gamma = \frac{2}{1-p_\mathrm{I}}$ et $\theta = 1 - \frac{4p_\circledast - 2}{1-p_\mathrm{I}}$.

Automates de Parikh non ambigus

En informatique, un mot est une suite finie de lettres. On appelle langage un ensemble de mots formés sur un ensemble fini fixé de lettres, appelé alphabet, souvent noté Σ. La théorie des langages s'intéresse particulièrement aux langages qui peuvent être décrits par des structures mathématiques finies, comme par exemple des automates (automates finis, automates à pile, machines à compteurs), des règles de réécriture (grammaires algébriques), ou encore d'autres modèles plus complexes de machines (machine de Turing).

Tous ces modèles peuvent être considérés comme des représentations mathématiques de programmes ou d'ordinateurs, avec des restrictions supplémentaires portant par exemple sur la taille de leur mémoire, sur leur aptitude à modifier leur entrée, ou

sur les opérations arithmétiques qu'elles sont capables d'effectuer. Intuitivement, un modèle avec des restrictions très fortes reconnaîtra des langages peu complexes, mais aura de nombreuses propriétés intéressantes décidables, avec une complexité raisonnable ; à l'inverse, un modèle trop puissant reconnaîtra des langages plus riches, mais dont la plupart des problèmes algorithmiques associés auront une forte complexité, voire seront indécidables.

C'est dans cette problématique de classification des langages et des problèmes algorithmiques associés qu'a été introduite en 1956 la hiérarchie de Chomsky [4], qui incluait les langages réguliers, algébriques, contextuels et récursivement énumérables ; cette hiérarchie a ensuite été précisée et enrichie par de nombreux modèles d'automates, et n'est plus aujourd'hui linéaire (dans le sens où plusieurs classes de langages ne sont pas comparables pour l'inclusion).

Lorsqu'un modèle d'automate ou de machine est introduit, il est donc fondamental de tenter de l'insérer le plus précisément dans cette hiérarchie, en le comparant aux autres modèles existants, afin de saisir son expressivité et ses limites, et étudier ses propriétés algorithmiques. Dans cette recherche de compréhension d'un modèle, deux sous-classes importantes interviennent généralement : les versions déterministes et non ambiguës.

Les variantes déterministes d'un modèle sont constituées d'automates dont le comportement à tout instant est entièrement déterminé par l'état courant et la lettre en cours de lecture. Elles jouissent en général de plus nombreuses propriétés de clôture (comme l'union ou le complémentaire) et de meilleures bornes algorithmiques. Mais le déterminisme est une restriction très forte qui diminue en général beaucoup le pouvoir d'expression du modèle et sa concision.

En toute généralité, un automate est dit *non ambigu* si tout mot qu'il reconnaît admet un unique calcul acceptant. Les automates non ambigus peuvent être vus comme un bon compromis à mi-chemin entre les versions déterministes trop rigides, et les versions non déterministes trop complexes à étudier : ils reconnaissent ainsi plus de langages que les variantes déterministes, tout en conservant certaines propriétés décidables. Par exemple, l'universalité qui est indécidable pour les langages algébriques, devient décidable pour les langages algébriques non ambigus.

La non-ambiguïté est une notion particulièrement difficile à comprendre car c'est une contrainte externe au modèle, qui porte sur le comportement global de l'automate, contrairement au déterminisme qui est une condition locale sur les règles de transitions. Pour un modèle d'automate donné, les langages de ce modèle qui ne sont reconnaissables par aucun automate non ambigu sont appelés *intrinsèquement ambigus*. Ces langages sont très intéressants pour comprendre les limites d'expressivité du modèle étudié, et séparer la classe générale de sa sous-classe non ambiguë. Malheureusement, pour de nombreux modèles (comme les langages algébriques ou les langages reconnus par des automates de Parikh), décider l'intrinsèque ambiguïté d'un langage est indécidable [10, 2].

En 1961, Parikh démontre pour la première fois qu'il existe des langages algébriques intrinsèquement ambigus [13], et ouvre la voie à l'élaboration de plusieurs techniques, par itération, qui ont permis d'enrichir les exemples de langages algébriques intrinsèquement ambigus. Malgré ces avancées, les méthodes développées, restreintes à des langages assez simples, sont très spécifiques et s'adaptent mal à des variations, même simples, des langages étudiés. Par ailleurs, elles ne s'appliquent qu'aux langages algébriques, et ne semblent pas pouvoir se généraliser à d'autres modèles de langages.

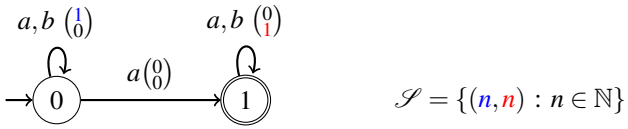
À tout langage $\mathscr{L}$, on peut associer sa série génératrice $L(z) := \sum_{n\in\mathbb{N}} \ell_n z^n$, où ℓ_n désigne le nombre de mots du langage $\mathscr{L}$ de taille n. Le théorème de Chomsky-Schützenberger [5] établit que tout langage algébrique non ambigu possède une série génératrice algébrique, c'est-à-dire qu'il existe un polynôme $P(z,Y) \in \mathbb{Q}[z,Y]$ tel que $P(z,L(z)) = 0$. En 1987, Philippe Flajolet [7] revisite ce théorème et développe une nouvelle méthode pour aborder l'intrinsèque ambiguïté. Il démontre l'intrinsèque ambiguïté de nombreux langages algébriques, en montrant que leur série génératrice n'est pas algébrique. Son approche est complémentaire aux techniques par itération évoquées précédemment. Surtout, elle est rapide et bien plus systématique pour aborder le problème de l'intrinsèque ambiguïté des langages algébriques.

Démarche. Depuis l'article de Philippe Flajolet [7], de nombreuses classes de langages ont été introduites en théorie des langages et en vérification. En particulier, les machines à compteurs sont des extensions maintenant bien comprises des langages réguliers et des langages algébriques. Du côté des séries, il existe également une généralisation des séries algébriques, les séries holonomes, qui sont un sujet actif de recherche en calcul formel et combinatoire.

Dans ma thèse, j'ai actualisé le lien entre théorie des automates et calcul formel à la lumière de ces avancées. J'ai ainsi transposé la démarche de Flajolet dans le monde des automates à compteurs (machines à compteurs à inversion bornée et automates de Parikh, avec ou sans pile).

Un automate de Parikh de dimension d est un automate fini dont les transitions sont étiquetées par une lettre et un vecteur de dimension d, auquel on associe un ensemble semilinéaire $\mathscr{S} \subseteq \mathbb{N}^d$, c'est-à-dire un ensemble de vecteurs qui peut être décrit par une formule de la logique du premier ordre sur les entiers naturels munis de l'addition. Un mot est accepté par l'automate s'il étiquette un calcul de l'automate d'un état initial à un état final, et si la somme des vecteurs des transitions empruntées appartient à l'ensemble $\mathscr{S}$.

Un automate de Parikh est dit non ambigu si tout mot est accepté par au plus un calcul acceptant. L'automate ci-dessous reconnaît le langage des mots de longueur impaire dont la lettre du milieu est un a. Cet automate est non ambigu, mais n'est pas déterministe.



Une série $L(z)$ est dite *holonome* ou *D-finite* si elle vérifie une équation différentielle linéaire à coefficients polynomiaux, autrement dit s'il existe un ordre r, et des polynômes $p_0, \ldots, p_r$ dans $\mathbb{Q}[z]$ tels que

$$p_r(z)\frac{d^r}{dz^r}L(z) + \ldots + p_0(z)L(z) = 0\,.$$

Contributions. Dans ma thèse, j'ai montré que la série génératrice d'un langage reconnu par un automate de Parikh (à pile) non ambigu est holonome. J'ai précisément inséré la classe des automates de Parikh non ambigus (avec ou sans pile) dans le paysage des langages formels, en la reliant notamment aux machines à compteurs à inversion bornée non ambiguës, avec ou sans pile.

De même, j'ai précisé plus finement les séries atteintes par les langages reconnus par des automates de Parikh (respectivement à pile) non ambigus à l'intérieur de la classe des séries holonomes : il s'agit exactement des diagonales de séries $\mathbb{N}$-rationnelles [9] (respectivement de séries $\mathbb{N}$-algébriques [1]). Ces travaux étendent ainsi l'approche de Philippe Flajolet, et permettent de démontrer l'intrinsèque ambiguïté de plusieurs langages de Parikh, en montrant que leur série génératrice n'est pas holonome.

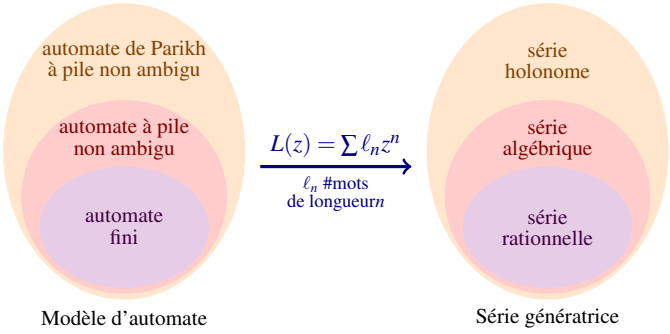


FIGURE 6. Schéma récapitulatif de la correspondance entre séries génératrices et langages non ambigus

Enfin, je me suis intéressé au problème de l'inclusion $L_{\mathscr{A}} \subseteq L_{\mathscr{B}}$ de deux automates de Parikh non ambigus $\mathscr{A}$ et $\mathscr{B}$. Ce problème est naturel dans le contexte de la vérification, où $\mathscr{A}$ est l'automate étudié, et l'automate $\mathscr{B}$ décrit les comportements

que l'on autorise ; vérifier que le langage de $\mathscr{A}$ est inclus dans celui de $\mathscr{B}$, c'est vérifier que les comportements décrits par $\mathscr{A}$ restent autorisés. Si le problème de l'inclusion est indécidable pour les automates de Parikh, il devient décidable pour les automates de Parikh non ambigus.

Ainsi, en utilisant l'algorithmique des séries holonomes, j'ai obtenu une première borne doublement exponentielle sur la taille du plus petit mot témoin de la stricte inclusion de deux automates de Parikh non ambigus. J'en ai déduit un algorithme de complexité doublement exponentielle pour résoudre le problème de l'inclusion pour les automates de Parikh non ambigus. Pour établir cette borne, j'ai notamment étudié minutieusement l'algorithme historique de Lipshitz [11], qui démontre que les séries holonomes sont closes par diagonale et produit d'Hadamard.

Références

[1] Cyril Banderier and Michael Drmota. Formulae and asymptotics for coefficients of algebraic functions. *Combinatorics, Probability and Computing*, 24(1) :1–53, 2015.

[2] Alin Bostan, Arnaud Carayol, Florent Koechlin, and Cyril Nicaud. Weakly-unambiguous parikh automata and their link to holonomic series. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 114 :1–114 :16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[3] Sabine Broda, António Machiavelo, Nelma Moreira, and Rogério Reis. Average size of automata constructions from regular expressions. *Bulletin of the EATCS*, 116, 2015.

[4] Noam Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3) :113–124, 1956.

[5] Noam Chomsky and Marcel-Paul Schützenberger. *The Algebraic Theory of Context-Free Languages*, volume 35 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1963.

[6] Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu. Spot 2.0 — a framework for LTL and ω-automata manipulation. In *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA'16)*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129. Springer, October 2016.

[7] Philippe Flajolet. Analytic models and ambiguity of context-free languages. *Theor. Comput. Sci.*, 49(2) :283 – 309, 1987.

[8] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[9] Scott Garrabrant and Igor Pak. Counting with irrational tiles. *arXiv preprint arXiv :1407.8222*, 2014.

[10] Seymour Ginsburg and Joseph Ullian. Ambiguity in context free languages. *J. ACM*, 13(1) :62–89, 1966.

[11] Leonard Lipshitz. The diagonal of a D-finite power series is D-finite. *J. Algebra*, 113(2) :373 – 378, 1988.

[12] Cyril Nicaud. On the Average Size of Glushkov's Automata. In Adrian-Horia Dediu, Armand-Mihai Ionescu, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications, Third International Conference, LATA 2009, Tarragona, Spain, April 2-8, 2009. Proceedings*, volume 5457 of *Lecture Notes in Computer Science*, pages 626–637. Springer, 2009.

[13] Rohit J Parikh. Language generating devices. *Quarterly Progress Report*, 60 :199–212, 1961.

[14] Heikki Tauriainen. Automated testing of Büchi automata translators for linear temporal logic. Research Report A66, Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland, December 2000.