RAPPORT SUR LES TRAVAUX ANTÉRIEURS

LÊ THÀNH DŨNG (TITO) NGUYỄN – CONCOURS CR CNRS 2024

1. Thématiques de recherche globales

Mes recherches portent principalement[1] sur de l'informatique théorique, plus précisément sur des aspects liés à la logique, la vérification formelle ou la sémantique de programmes – c'est-à-dire les sujets abordés dans le Volume B du *Handbook of Theoretical Computer Science*. Deux de mes sujets principaux de recherche sont la *logique linéaire* et les *transducteurs*. Ce sont des objets d'étude classiques de deux courants assez distincts du « Volume B » : respectivement la théorie des langages de programmation et la théorie des automates. Ainsi, mes travaux cherchent en grande partie à *relier entre eux différents pans de l'informatique théorique* (y compris le « Volume A », voir plus loin).

λ-calcul et logique linéaire. Via la *correspondance preuves-programmes* (ou « correspondance de Curry–Howard »), divers systèmes de preuves formelles s'avèrent être isomorphes à des langages de programmation théoriques. C'est par exemple le principe de base de l'assistant de preuve Coq, qui combine programmation et certification dans un même cadre.

La *logique linéaire* [Gir87] est issue de cette correspondance. Côté programmation, les systèmes de types linéaires contrôlent l'usage de la duplication ; des idées semblables interviennent au sein du langage de programmation système Rust pour contrôler des ressources. Notons que Rust s'inspire en grande partie de l'expérience passée des langages fonctionnels fortement typés (Haskell, OCaml, ...) dont les modèles théoriques idéalisés sont des *λ-calculs typés*.

Un de mes thèmes de recherche principaux est d'étudier le *pouvoir calculatoire* de λ-calculs à typage linéaire. Et du côté « preuves » de la correspondance preuves-programmes, certains de mes travaux portent sur la *théorie de la démonstration structurelle*, étudiant la combinatoire de systèmes de preuves formelles pour des variantes de la logique linéaire.

Transducteurs. La théorie des automates s'intéresse à des modèles de calcul « à états finis » et donc suffisamment restreints pour qu'on sache les exécuter efficacement, ce qui peut présenter un intérêt pour traiter de gros volumes de données. Souvent, ces automates servent à reconnaître des langages ; autrement dit, ils calculent des fonctions à sortie booléenne. On parle de *transducteurs* pour des automates à sortie plus riche – typiquement, calculant une fonction des mots vers les mots. Les transducteurs contemporains sont en fait plus expressifs qu'on pourrait s'y attendre[2] ; par exemple, la fonction ci-dessous (où $\#$ est une lettre servant de séparateur), appelée « inner squaring », est calculable par un transducteur à jetons (cf. §4) :

$$w_0\# \dots \#w_n \mapsto (w_0)^n\# \dots \#(w_n)^n \qquad (\text{par ex. } aa\#ba\#b \mapsto aaaa\#baba\#bb)$$

La théorie des transducteurs compare donc les classes de fonctions définies par ces modèles de calcul, par des équivalences, inclusions, séparations... ou parfois par des algorithmes pour des problèmes d'appartenance à une sous-classe $\mathcal{C}' \subseteq \mathcal{C}$: « étant donnée une fonction $f \in \mathcal{C}$ en entrée,

[1] J'ai aussi travaillé pendant 5 mois comme ingénieur de recherche sur l'assistant de preuve Squirrel pour protocoles de sécurité (`https://squirrel-prover.github.io/`). Mes contributions sur ce projet – principalement, l'ajout d'une intégration avec des solveurs SMT (Satisfiability Modulo Theories) – étant très modestes et relativement déconnectées du reste de mes travaux, je ne les détaillerai pas ici.

[2] Citons un *survey* récent de Bojańczyk [Boj22] : « In later years, transducers have grown in sophistication, reaching a point where they resemble programming languages, with constructs such as numerical variables, loops, recursion or higher-order functions. This means that modern models of transducers can now be actually useful programs, as opposed to being idealized and radically simplified models of useful programs. All of this while retaining the good decidability properties of automata (transducers are not supposed to be Turing complete, which is framed here as an advantage : the halting problem is decidable) and their attractive mathematical theory. »

f est-elle en fait dans $\mathcal{C}'$? ». Ces problèmes sont le plus souvent indécidables sur des programmes quelconques, mais on peut espérer trouver des algorithmes pour certains modèles d'automates ou de transducteurs. Ceci s'applique plus généralement à des problèmes qui s'apparentent à de l'analyse statique, et c'est là un intérêt majeur de se restreindre à un cadre « à états finis ».

Liens avec l'informatique théorique « Volume A ». Je me suis aussi intéressé (de façon moins centrale) à l'algorithmique, la combinatoire et la théorie de la complexité. Ainsi, j'ai suivi durant ma formation initiale un M2 en recherche opérationnelle. Quelques-uns de mes travaux (§5–§6) consistent à étudier la complexité de problèmes ou à caractériser des classes de complexité, et certains s'appuient sur une connaissance de la littérature en algorithmique des graphes (§6). Enfin, je participe à (et ai été employé comme post-doctorant sur) un projet ANR se situant à l'interface entre λ-calcul et combinatoire (`https://www.lix.polytechnique.fr/LambdaComb/`).

2. Un programme de recherche en cours : automates implicites

Complexité implicite. La correspondance preuves-programmes implique souvent des langages de programmation qui garantissent que les programmes *terminent* car cela implique, côté preuves formelles, la cohérence logique. Ainsi, le λ-calcul simplement typé ajoute par-dessus du λ-calcul pur (exemple historique de langage Turing-complet) un système de typage ; on peut alors exclure la possibilité de non-terminaison en ne gardant que les programmes « bien typés ».

Une telle garantie de terminaison peut même venir avec des bornes quantitatives de complexité en temps. Ainsi, par exemple, Hillebrand et al. [HKM96] montrent que les programmes en λ-calcul simplement typé opérant sur des codages de données astucieux peuvent calculer précisément les fonctions dans la classe ELEMENTARY. Ce résultat est typique d'un domaine appelé *complexité implicite* (voir [Péc20]), qui cherche à caractériser des classes de complexité en employant des langages « de haut niveau », sans mentionner explicitement des bornes en temps ou en espace.

La duplication des données étant une source majeure de complexité algorithmique, son contrôle au moyen de variantes de la logique linéaire a donné lieu à de nombreux travaux de complexité implicite, à commencer par les caractérisation du temps polynomial (P) dans [GSS92 ; Gir98].

Le théorème de Hillebrand et Kanellakis. Une inspiration centrale de mes recherches est :

Théorème 1 ([HK96])**.** *Les fonctions des mots vers les booléens*[3] *définissables en* λ-calcul simplement typé *en utilisant une représentation « naïve » des entrées* (codage de Church des mots) *correspondent exactement aux* langages rationnels (ou réguliers)*, c'est-à-dire au pouvoir des automates finis.*

Il se démarque au sein de la complexité implicite par sa naturalité : alors qu'il est habituel de concevoir un nouveau langage de programmation *ad hoc* ou pour obtenir une correspondance avec une classe de complexité ciblée à l'avance, ici, on voit les automates apparaître « sans faire exprès » en posant une question d'expressivité sur un λ-calcul préexistant !

Vers les automates implicites. Le Théorème 1 suggère d'établir, via les codages de Church, des liens entre le pouvoir expressif de λ-calculs typés « naturels » et de modèles d'automates (une idée semblable existait aussi dans le domaine connexe du *model-checking d'ordre supérieur*). Avec **Cécilia Pradic**, nous avons initié un programme de recherche sur cette piste de la « complexité implicite pour les automates » ; c'est devenu le sujet central de mon manuscrit de thèse[4] [Ngu21].

- Notre premier résultat [NP20] concerne les *langages sans étoile*. Ils forment une sous-classe célèbre des langages rationnels, qui admet de nombreuses définitions équivalentes entre elles (expressions régulières sans étoile, logique du premier ordre, etc.). Nous sommes parvenus à la caractériser dans un λ-calcul muni d'un *typage non commutatif* : on peut forcer les arguments d'une fonction à être utilisés dans l'ordre où ils lui sont passés.

  La non-commutativité nécessite que le système de types soit linéaire ou affine ; l'idée remonte à un travail de Lambek [Lam58] aux motivations linguistiques, précurseur de la logique linéaire. Des liens avec les cartes combinatoires sont également connus [ZG15].

[3] Subtilité : on autorise une substitution dans le type d'entrée, voir [Ngu21, Th. 1.1.3] et la discussion qui suit.

[4] Dont la plupart des résultats n'ont pas été publiés officiellement ailleurs.

Notre résultat est cependant, à ma connaissance, la première conséquence *calculatoire* non-triviale d'un système de types non-commutatif (en le rendant commutatif, on obtient les langages rationnels au lieu des langages sans étoile) à être découverte.

- En utilisant un système de types linéaire (mais commutatif), nous caractérisons deux classes de fonctions entre mots calculées par des transducteurs [Ngu21, Theorem 1.2.3]. Une de ces deux classes – celle des *fonctions régulières* – peut être définie par des *streaming string transducers* (SST) restreints à être *sans copie* [AČ10]. Cette condition analogue au typage linéaire porte sur la manipulation des registres constituant la mémoire des SST.

  Nous traitons également les fonctions régulières entre arbres, en remarquant que la différence entre « sans copie » et des formes de « single use restriction » plus permissives (et moins aisées à décrire) de certains transducteurs d'arbres [EM99 ; AD17] peut s'exprimer par les connecteurs additifs de la logique linéaire.[5]

Une caractérisation des fonctions régulières d'arbres assez proche[6] de la nôtre a été obtenue de façon indépendante par Gallot, Lemay et Salvati [GLS20]. Elle repose sur des transducteurs qui stockent des λ-termes en mémoire. **Gabriele Vanoni** [NV24] et moi avons récemment étendu cette approche « λ-transducteurs » pour caractériser encore d'autres classes de fonctions entre arbres et pour résoudre une question ouverte que Pradic et moi avions posée dans [NP20]. Cela dit, notre second résultat avec Pradic sur les fonctions entre mots, dont nous reparlerons en section 4, ne rentre pas dans ce cadre « λ-transducteurs ».

3. Quelques interventions de la théorie des catégories

Pour démontrer les liens susmentionnés entre transducteurs et λ-calcul, Cécilia Pradic et moi avons été amenés à utiliser le cadre mathématique unificateur des *catégories*. Plus généralement, les catégories sont apparues dans plusieurs de mes travaux sous deux aspects : théorie catégorique des automates, et sémantique catégorique (dénotationnelle) des langages de programmation.

(J'enseigne également une introduction aux catégories en M2 informatique à l'ÉNS Lyon.)

Évaluation sémantique et langages rationnels de λ-termes. La démonstration du Théorème 1 par Hillebrand et Kanellakis [HK96] utilise une *sémantique dénotationnelle* interprétant les types simples comme ensembles finis et les λ-termes (programmes) comme fonctions sur ces ensembles. À partir d'un λ-terme définissant une fonction des mots vers les booléens, on déduit l'ensemble fini d'états d'un automate déterministe, et ses fonctions de transition. La réciproque – traduire un automate en λ-terme – est un exercice de programmation assez simple.

Indépendamment, Salvati [Sal09 ; Sal15] a utilisé la sémantique dans les ensembles finis pour définir une notion de langage rationnel de λ-termes simplement typés, qui a été reprise dans des travaux ultérieurs [Mel17 ; GMM23]. Pour les codages de Church de mots, on retrouve les langages rationnels habituels. Avec **Vincent Moreau** [MN23], nous montrons que cette notion est robuste :

- Nous proposons une caractérisation syntaxique de ces langages rationnels de λ-termes. Un sens de l'équivalence étend l'argument de Hillebrand et Kanellakis ; la réciproque est bien moins triviale que dans le cas des mots, et utilise un argument de « relations logiques ».
- Nous montrons également qu'on peut remplacer les ensembles finis par de nombreuses autres sémantiques dénotationnelles dans la définition de [Sal09], et toujours obtenir la même classe de langages. Ces sémantiques sont données par des *catégories cartésiennes closes* avec une condition supplémentaire exprimant un caractère finitaire.

Cécilia Pradic et moi avons aussi recours à un argument semblable à [HK96] (mais plus élaboré) pour démontrer nos résultats « d'automates implicites » concernant les transducteurs (voir §2). Tout d'abord, on représente une variante des *streaming string transducers* en utilisant une catégorie $\mathcal{SR}_{\oplus\&}$, suivant une définition générale [CP20] « d'automate sur une catégorie ». On exhibe ensuite une structure de *catégorie monoïdale close* sur $\mathcal{SR}_{\oplus\&}$ pour y interpréter notre λ-calcul typé.

[5] Les résultats mentionnés dans cet item n'ont pas été publiés en conférence ou en journal (une soumission à LMCS a été rejetée pour longueur excessive) mais constituent une portion significative de mon manuscrit de thèse.

[6] Une différence majeure est l'usage de *regular lookahead* au lieu de connecteurs additifs chez [GLS20].

Théorie catégorique des automates. Le travail susmentionné de Colcombet et Petrişan [CP20] fait partie d'un courant de recherches généralisant de façon abstraite des constructions et résultats classiques de théorie des automates, afin de les rendre plus largement applicables. Dans cet esprit, nous démontrons aussi quelques résultats [Ngu21, Ch. 4] (toujours avec **Cécilia Pradic**) visant à montrer que la notion de catégorie monoïdale close a une pertinence pour la théorie des automates qui dépasse le simple lien avec le λ-calcul linéaire. Intuitivement, cela est dû au rôle important joué par des « espaces de fonctions » dans les constructions sur les automates : typiquement, quand on exploite la finitude de $Q \to Q$ pour un ensemble fini Q d'états.

Passer de Q à $Q \to Q$, c'est aussi passer d'un automate à un *semigroupe*. Une des définitions classiques des langages rationnels est la reconnaissance par semigroupes finis ; nos résultats sur les langages sans étoile (§2) reposent sur leur caractérisation par les semigroupes apériodiques (et la décomposition de Krohn–Rhodes de tels semigroupes). Avec **Mikołaj Bojańczyk** [BN23], nous avons proposé une caractérisation algébrique analogue, et particulièrement concise, des *fonctions régulières* sur les mots précédemment évoquées. Là où la reconnaissance algébrique d'un langage rationnel (ou sans étoile) fait intervenir un morphisme de semigroupes, nous utilisons en plus un *foncteur* de la catégorie des semigroupes vers elle-même pour reconnaître une fonction régulière.

L'esprit de ce dernier travail [BN23] est assez différent des travaux de théorie catégorique des automates comme [CP20] : il ne s'agit pas de reformuler des constructions connues dans un cadre plus général, mais d'introduire une nouvelle caractérisation de la classe concrète des fonctions régulières, dont l'équivalence avec les définitions précédentes est non triviale.

4. Fonctions polyrégulières

Transducteurs à jetons (sans comparaison). Mon travail sur les automates implicites (§2) avec Pradic faisait apparaître deux définitions naturelles de classes de fonctions sur les mots calculées par des λ-termes à typage linéaires. L'une d'entre elles donne les fonctions régulières que nous venons de mentionner, et qui ont été intensément étudiées depuis une vingtaine d'années (voir [FR16 ; MP19]). Par contre, à notre connaissance, l'autre n'existait pas encore dans la littérature.

Elle s'est néanmoins révélée naturelle au vu des travaux récents de Bojańczyk et al. sur les *fonctions polyrégulières* [Boj18 ; BKL19 ; Boj22]. Ces dernières peuvent être par les « transducteurs à jetons » (*pebble transducers*), qui examinent leur entrée avec plusieurs têtes de lecture (appelées jetons) bidirectionnelles. En ôtant à ces machines la capacité à *comparer les positions de leurs jetons* on caractérise en fait les fonctions survenues dans nos recherches sur le λ-calcul linéaire.

Ceci nous a motivé à étudier cette classe des *fonctions polyrégulières sans comparaison* du point de vue de la théorie des automates « pure », dans l'article[7] [NNP21]. Nous y établissons quelques caractérisations alternatives, des propriétés souhaitables – notamment, nous avons découvert la clôture par composition grâce au lien avec le λ-calcul, mais en fournissons une preuve *self-contained* sans λ-termes – ainsi que des résultats de séparation avec d'autres classes.

Vitesse de croissance et optimisation. Notre démonstration la plus technique dans [NNP21] est un théorème de *minimisation des jetons* : si une fonction polyrégulière sans comparaison f vérifie $|f(w)| = O(|w|^k)$, alors elle peut être calculée par un transducteur à k jetons. (La réciproque est une observation immédiate.) Ce théorème a été reprouvé et étendu par Gaëtan Douéneau-Tabot dans sa thèse [Dou23], qui contient aussi d'autres résultats sur les polyrégulières sans comparaison.

Notre approche réutilise des outils développés par Nathan Lhote dans l'article [Lho20], qui affirme démontrer la minimisation des jetons pour les polyrégulières générales. L'article s'avère être irréparablement faux[8] : Bojańczyk [Boj23] a montré que pour tout $k \in \{3, 4, \dots\}$, on peut trouver une fonction polyrégulière à croissance[9] $O(|w|^2)$ nécessitant au moins k jetons.

[7] Co-signé avec l'autrice *fictionnelle* Camille Noûs : `https://www.cogitamus.fr/camille.html`

[8] J'ai également trouvé un théorème faux – mais moins central – dans l'article [BD20] de Mikołaj Bojańczyk et Amina Doumane paru à la même conférence LICS'20, comme signalé sur la version arXiv : `https://arxiv.org/abs/2002.09307`

[9] Les exemples à croissance quadratique sont minimaux : un corollaire du théorème principal de [EIM21] est que les fonctions polyrégulières à croissance linéaire sont des transductions MSO, qui dans le cas des mots sont calculées par des transducteurs à un jeton [EH01] – en fait, ce sont les fonctions régulières.

Douéneau-Tabot et moi sommes co-crédités dans [Boj23, Footnote 5] pour la découverte du contre-exemple pour $k = 3$, qui n'est autre que la fonction « inner squaring » de la section 1. Cette découverte s'est produite au sein d'une conversation par mail incluant également **Sandra Kiefer** et (encore) **Cécilia Pradic**. Dans [KNP23], Kiefer, Pradic et moi avons proposé une réfutation alternative de la minimisation des jetons, démontrant plus rapidement et moins techniquement (par réduction à de vieux résultats puissants sur les transducteurs d'arbres) un résultat légèrement plus fort que celui de Bojańczyk (séparant les *langages de sortie* de fonctions).

Bojańczyk prouve également dans [Boj23] un résultat positif reliant vitesse de croissance et appartenance à une sous-classe – dont il découle notamment qu'on peut calculer, à partir d'une fonction f polyrégulière, le ℓ minimal tel que $|f(w)| = O(|w|^\ell)$. Dans un travail en cours avec[10] **Nathan Lhote** et **Paul Gallot** [GLN24], nous démontrons de façon plus simple (pompage sur les automates pondérés, sans besoin des forêts de factorisation) une généralisation de ce résultat (aux « interprétations MSO ensemblistes » [CL07] des arbres vers des structures arbitraires). Grâce à cette généralité, nous espérons aussi reprouver des résultats réputés difficiles sur la croissance des transducteurs d'arbres [EIM21].

5. Problèmes ouverts résolus grâce aux automates implicites

Si j'ai commencé cette notice en parlant de *theory-building* (§2), les résultats ci-dessus au sujet de la croissance des fonctions polyrégulières illustrent plutôt mon approche générale du *problem-solving* : elle consiste essentiellement à puiser dans la littérature préexistante la bonne technologie permettant de trivialiser le problème. Dans les cas précédents, cela se faisait de façon interne à la théorie des automates. Les travaux que je cite dans cette section visent à montrer que le point de vue des automates implicites se révèle utile pour appliquer cette méthode.

Expressivité du λ-calcul élémentaire affine polymorphe. Il s'agit d'une question soulevée en conclusion d'un article de Patrick Baillot [Bai15][11] sur la complexité implicite en logique linéaire (cf. §2). Cet article donne une caractérisation de P (temps polynomial) reposant sur un λ-calcul dont le système de types autorise des types récursifs. Peut-on se passer de cette fonctionnalité ? La réponse est non : sans récursivité dans les types, le système obtenu, appelé *λ-calcul élémentaire affine polymorphe* dans mon article [Ngu19c], calcule en fait les langages rationnels et non P.

Ma preuve s'inspire très largement de celle du Théorème 1 par Hillebrand et Kanellakis. Des arguments syntaxiques réduisent le problème à l'existence d'une sémantique finitaire pour un sous-système. Dans un brouillon inachevé [Ngu19a], j'esquisse une démonstration de finitude pour une sémantique connue ; mais en fait, Jim Laird [Lai13] avait déjà construit une sémantique pour le sous-système qui nous intéresse dont il constate explicitement le caractère finitaire.

C'est ce premier succès – antérieur aux travaux cités en §2 – qui m'a convaincu de l'importance du théorème de Hillebrand et Kanellakis, et donc de me lancer dans le programme de recherche des « automates implicites ». Avec **Cécilia Pradic**, nous avons aussi brièvement exploré l'usage d'une variante du λ-calcul élémentaire affine polymorphe pour caractériser d'autres classes de complexité sous-polynomiales. Ceci a mené aux résultats partiels de l'article [NP19], qui eux aussi utilisent des méthodes sémantiques ; bien que publié à ICALP, cet article me paraît désormais peu concluant (et souffre d'un manque de rigueur).

Complexité de la convertibilité dans des λ-calculs. Un thème étroitement lié à la complexité implicite est l'étude de la complexité, pour divers λ-calculs, du problème de *convertibilité* suivant : « étant donnés deux λ-termes t et u, ont-ils la même forme normale ? ». Quand t est de type booléen et $u = \texttt{true}$, cela revient à *évaluer* le résultat renvoyé par t.

En λ-calcul simplement typé, on sait que ce problème est TOWER-complet [Sta79 ; Sch16]. Par contre, les preuves connues de ce résultat ne s'adaptaient pas au cas où on impose une condition supplémentaire de « sûreté » sur les λ-termes ; Blum et Ong [BO09, §3] ont analysé les obstructions

[10] Collaboration stimulée par notre participation commune au séminaire Dagstuhl sur les transformations régulières en mai 2023 : `https://www.dagstuhl.de/en/seminars/seminar-calendar/seminar-details/23202`.

[11] Je me suis en fait appuyé sur une reformulation ultérieure de [Bai15] plus *user-friendly* dans [BDR18].

à cela et conjecturé que cette restriction rendrait le problème de convertibilité strictement moins difficile. La meilleure borne inférieure qu'ils ont parvenus à obtenir est la PSPACE-difficulté.

J'ai réfuté cette conjecture [Ngu23], en démontrant que la convertibilité restreinte aux λ-termes sûrs (*safe*) reste TOWER-complète. Ma preuve est complètement différente de [Sta79] : c'est une réduction depuis le problème d'équivalence des expressions sans étoile. Elle utilise une traduction inductive, inspirée des « automates implicites », d'une expression vers un λ-terme reconnaissant le même langage. Par exemple, le traitement de la concaténation dans les expressions sans étoile s'appuie entre autres sur un λ-terme sûr calculant une fonction polyrégulière bien choisie.

Avec **Anupam Das**, **Damiano Mazza** et **Noam Zeilberger** [Das+23], nous avons également démontré la P-complétude de la convertibilité en λ-calcul planaire (un système proche du λ-calcul non-commutatif utilisé dans [NP20] pour caractériser les langages sans étoile, cf. §2).

Automates bidirectionnels à comportement planaire. Dans un exposé [Hin06], Peter Hines avait proposé un modèle d'automates sans caractériser son expressivité. Son modèle étant relié de près à une sémantique d'un λ-calcul linéaire non commutatif [Hin03 ; Abr07], mes travaux précédents [NP20] avec Pradic nous ont mené à conjecturer, puis à prouver, que ces automates reconnaissent précisément les langages sans étoile [NNP23]. Nous caractérisons également dans cet article la classe de fonctions entre mots calculée par le modèle correspondant de transducteurs, et montrons qu'une condition de réversibilité n'affecte pas le pouvoir expressif.

6. Combinatoire des réseaux de preuve en logique linéaire

Cette dernière section fournit encore un exemple de l'usage de liens entre domaines distincts au sein de l'informatique théorique – cette fois-ci, il s'agit de l'algorithmique des graphes et la théorie de la démonstration – pour résoudre des questions concrètes.

Réseaux de preuve. L'un des premiers apports de la logique mathématique a été de voir les démonstrations elles-mêmes comme des objets mathématiques : naïvement, une liste de formules logiques dont chacune découle des précédentes. Il s'est vite révélé avantageux de concevoir des formalismes de preuve arborescents (calcul des séquents, déduction naturelle, ...) aux propriétés structurelles plus riches. L'article fondateur de logique linéaire [Gir87] va plus loin en proposant de représenter les preuves comme des sortes de *graphes* : les *réseaux de preuve*.

Ceux-ci présentent de nombreux avantages, mais posent un nouveau défi. La correction d'une preuve arborescente est vérifiable *localement* : il suffit de vérifier que tout nœud de l'arbre applique une règle logique valide. Dans un réseau de preuve, il faut combiner cette validité locale avec une propriété combinatoire *globale* nommée *critère de correction*. Dans le cas le plus simple, celui de la logique linéaire multiplicative (MLL), la situation est bien comprise : les réseaux corrects sont traduisibles en preuves de calcul des séquents et vice-versa, et le problème de correction est décidable en temps linéaire [Gue11] et NL-complet [JM11]. Surprenamment, ces résultats non triviaux sont les fruits d'une théorie combinatoire *ad hoc* des réseaux de preuve.

Couplages parfaits et réseaux MLL+Mix. La seule tentative de relier les réseaux de preuve à la théorie des graphes « mainstream » était le travail de Christian Retoré [Ret03, §1] réduisant la correction des réseaux MLL+Mix à un problème classique : vérifier qu'un couplage parfait donné dans un graphe est unique. (La logique MLL+Mix étend MLL avec une règle supplémentaire, et le problème de correction pour MLL se réduit à celui pour MLL+Mix [Ngu20, Remark 4.2].)

La réduction étant en temps linéaire, et ce dernier problème de couplages aussi [GKT01], on a en fait immédiatement un algorithme en temps linéaire pour la correction MLL+Mix – et donc pour MLL. Étrangement, ni ce raisonnement, ni même le résultat pour MLL+Mix, ne semblaient avoir été signalés dans la littérature avant mon article [Ngu20]. Peut-être pensait-on que le cas MLL+Mix était trop proche du cas MLL pour qu'il puisse y avoir de surprise, mais je fournis des contre-arguments à cette idée dans [Ngu20], notamment celui-ci : un algorithme NL pour la correction pour les réseaux MLL+Mix résoudrait du même coup une vieille conjecture de László Lovász, or les techniques employées pour le résultat analogue pour MLL [JM11] semblent trop faibles pour cela ; l'explication est qu'elles exploitent de la structure spécifique à MLL sans Mix.

En fait, les liens structurels entre la combinatoire des couplages parfaits et celle des réseaux MLL+Mix sont plus profonds que de simples réductions algorithmiques ; c'est déjà apparent dans l'article de Retoré [Ret03], et je donne aussi quelques résultats dans ce sens dans [Ngu20]. Par exemple, on peut reformuler une propriété sur les « dépendances logiques » dans les réseaux de preuve en théorème équivalent sur les couplages, susceptible d'intéresser un public plus large.[12]

Logique pomset $\neq$ système BV. On doit également à Retoré la *logique pomset* [Ret97], étendant MLL+Mix avec un connecteur non-commutatif intermédiaire entre « et » et « ou ». Cette logique se formule facilement en réseaux de preuve, mais pas en calcul des séquents. Cela a motivé Alessio Guglielmi à introduire l'*inférence profonde*[13] – un paradigme où les preuves peuvent être vérifiées localement mais qui permet plus de liberté que les séquents – et à s'en servir comme base du *système BV* [Gug07]. Ce dernier étend aussi MLL+Mix avec un connecteur non-commutatif, et l'équivalence entre BV et la logique pomset était même conjecturée depuis 2 décennies [GS01].

Lutz Straßburger et moi avons *réfuté cette conjecture* [NS23]. Ce travail est en fait en continuité directe avec le précédent : son point de départ est que les mêmes outils de théorie des graphes permettent de montrer que la correction des réseaux pomset est coNP-complète[14] (en s'inspirant de travaux sur les digraphes arête-coloriés [Gou+13]), et que la prouvabilité en logique pomset est Σ_2^{p}-complète. Ce serait strictement plus difficile que la prouvabilité en système BV – qui est « seulement » NP-complète – si l'on suppose NP $\neq$ coNP. Cela donnait dans un premier temps une réfutation conditionnelle ; nous avons ensuite trouvé un contre-exemple explicite. L'article [NS23] contient quelques autres résultats autour de ces deux logiques.

Géométrie de l'interaction. Une autre contribution fondamentale de Retoré est une version des réseaux MLL+Mix basée sur les *cographes*[15] [Ret03, §2]. Avec[16] **Thomas Seiller** [NS18], nous avons proposé une reconstruction du critère de correction pour ces réseaux cographiques à partir du *comportement calculatoire* de leur interprétation dans une certaine sémantique. Nous expliquons donc un aspect « preuve » par l'invocation des « programmes », ce qui en fait mon seul travail qui tire réellement parti de Curry–Howard. Au fond, nous avons surtout réactualisé de vieilles idées de Jean-Yves Girard [Gir89, §IV.5] sur la « géométrie de l'interaction », en utilisant une variante issue la thèse de Seiller [Sei12, §5.2.34].

Des incarnations plus contemporaines de la géométrie de l'interaction apparaissent aussi dans certains de mes travaux précédemment évoqués : sa version catégorique a inspiré [NNP23], et la « machine abstraite d'interaction » est utilisée crucialement dans [NV24].

Références

[Abr07] Samson Abramsky. « Temperley–Lieb Algebra : From Knot Theory to Logic and Computation via Quantum Mechanics ». en. In : *Mathematics of Quantum Computation and Quantum Technology*. Sous la dir. de Goong Chen, Louis Kauffman et Samuel Lomonaco. T. 20074453. Chapman et Hall/CRC, sept. 2007, p. 515-558. doi : 10.1201/9781584889007.ch15 (cf. p. 6).

[AČ10] Rajeev Alur et Pavol Černý. « Expressiveness of streaming string transducers ». In : *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*. Sous la dir. de Kamal Lodaya et Meena Mahajan. T. 8. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010, p. 1-12. doi : 10.4230/LIPIcs.FSTTCS.2010.1 (cf. p. 3).

[AD17] Rajeev Alur et Loris D'Antoni. « Streaming Tree Transducers ». en. In : *Journal of the ACM* 64.5 (août 2017), p. 1-55. issn : 00045411. doi : 10.1145/3092842 (cf. p. 3).

[AHS22] Matteo Acclavio, Ross Horne et Lutz Strassburger. « An Analytic Propositional Proof System on Graphs ». In : *Logical Methods in Computer Science* 18.4 (2022). doi : 10.46298/lmcs-18(4:1)2022 (cf. p. 7).

[12] Ce théorème est énoncé et prouvé dans une note [Ngu19b] à destination de théoriciens des graphes, où je comble aussi quelques omissions mineures dans la littérature sur les graphes que j'avais repérées. Étant donné la faible difficulté technique, je n'ai pas tenté de faire publier cette note, mais elle est arXivée à disposition de tous.

[13] *Deep inference*, un sujet de recherche actif, voir http://alessio.guglielmi.name/res/cos/

[14] Encore une erreur dans une publication : Retoré prétendait que ce problème était dans P [Ret97, Prop. 5].

[15] Elle a servi de base à de nombreux travaux ultérieurs, par exemple sur les « preuves combinatoires » [Hug06 ; HHS19]. Dans la conclusion de [NS18], nous suggérons que la décomposition modulaire permettrait d'aller au-delà des cographes, en considérant des graphes arbitraires comme formules généralisées ; cela s'est concrétisé notamment dans [AHS22].

[16] Valentin Maestracci a trouvé une erreur dans [NS18, démonstration de la Proposition 37].

[Bai15] Patrick Baillot. « On the expressivity of elementary linear logic : Characterizing Ptime and an exponential time hierarchy ». In : *Information and Computation* 241 (avr. 2015), p. 3-31. issn : 0890-5401. doi : 10.1016/j.ic.2014.10.005 (cf. p. 5).

[BD20] Mikołaj Bojańczyk et Amina Doumane. « First-order tree-to-tree functions ». In : *LICS '20 : 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany (online conference), July 8-11, 2020*. Sous la dir. d'Holger Hermanns, Lijun Zhang, Naoki Kobayashi et Dale Miller. ACM, 2020, p. 252-265. doi : 10.1145/3373718.3394785 (cf. p. 4).

[BDR18] Patrick Baillot, Erika De Benedetti et Simona Ronchi Della Rocca. « Characterizing polynomial and exponential complexity classes in elementary lambda-calculus ». In : *Information and Computation*. Developments in Implicit Computational Complexity (DICE) 2014 and 2015 261 (août 2018), p. 55-77. issn : 0890-5401. doi : 10.1016/j.ic.2018.05.005 (cf. p. 5).

[BKL19] Mikołaj Bojańczyk, Sandra Kiefer et Nathan Lhote. « String-to-String Interpretations With Polynomial-Size Output ». In : *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Sous la dir. de Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini et Stefano Leonardi. T. 132. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 106 :1-106 :14. isbn : 978-3-95977-109-2. doi : 10.4230/LIPIcs.ICALP.2019.106 (cf. p. 4).

[BN23] Mikołaj Bojańczyk et Lê Thành Dũng Nguyễn. « Algebraic Recognition of Regular Functions ». In : *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*. Sous la dir. de Kousha Etessami, Uriel Feige et Gabriele Puppis. T. 261. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 117 :1-117 :19. doi : 10.4230/LIPICS.ICALP.2023.117 (cf. p. 4).

[BO09] William Blum et C.-H. Luke Ong. « The Safe Lambda Calculus ». en. In : *Logical Methods in Computer Science* 5.1 (fév. 2009). doi : 10.2168/LMCS-5(1:3)2009 (cf. p. 5).

[Boj18] Mikołaj Bojańczyk. « Polyregular Functions ». In : *CoRR* abs/1810.08760 (oct. 2018). arXiv : 1810.08760 (cf. p. 4).

[Boj22] Mikołaj Bojańczyk. « Transducers of polynomial growth ». In : *LICS '22 : 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*. Sous la dir. de Christel Baier et Dana Fisman. ACM, 2022, 1 :1-1 :27. doi : 10.1145/3531130.3533326 (cf. p. 1, 4).

[Boj23] Mikołaj Bojańczyk. « On the Growth Rates of Polyregular Functions ». In : *38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2023. doi : 10.1109/LICS56636.2023.10175808 (cf. p. 4, 5).

[CL07] Thomas Colcombet et Christof Löding. « Transforming structures by set interpretations ». In : *Logical Methods in Computer Science* 3.2 (2007). doi : 10.2168/LMCS-3(2:4)2007 (cf. p. 5).

[CP20] Thomas Colcombet et Daniela Petrişan. « Automata Minimization : a Functorial Approach ». en. In : *Logical Methods in Computer Science* 16.1 (mars 2020). doi : 10.23638/LMCS-16(1:32)2020 (cf. p. 3, 4).

[Das+23] Anupam Das, Damiano Mazza, Lê Thành Dũng Nguyễn et Noam Zeilberger. *On the Complexity of Normalization for the Planar λ-calculus*. Presented at the 7th International Workshop on Trends in Linear Logic and Applications. 2023. url : https://lipn.univ-paris13.fr/TLLA/2023/abstracts/07-Final.pdf (cf. p. 6).

[Dou23] Gaëtan Douéneau-Tabot. « Optimization of string transducers ». en. Thèse de doct. Université Paris Cité, nov. 2023. url : https://gdoueneau.github.io/pages/DOUENEAU-TABOT_Optimization-transducers_v1.pdf (cf. p. 4).

[EH01] Joost Engelfriet et Hendrik Jan Hoogeboom. « MSO definable string transductions and two-way finite-state transducers ». en. In : *ACM Transactions on Computational Logic* 2.2 (avr. 2001), p. 216-254. issn : 15293785. doi : 10.1145/371316.371512 (cf. p. 4).

[EIM21] Joost Engelfriet, Kazuhiro Inaba et Sebastian Maneth. « Linear-bounded composition of tree-walking tree transducers : linear size increase and complexity ». In : *Acta Informatica* 58.1-2 (2021), p. 95-152. doi : 10.1007/s00236-019-00360-8 (cf. p. 4, 5).

[EM99] Joost Engelfriet et Sebastian Maneth. « Macro Tree Transducers, Attribute Grammars, and MSO Definable Tree Translations ». In : *Information and Computation* 154.1 (oct. 1999), p. 34-91. issn : 0890-5401. doi : 10.1006/inco.1999.2807 (cf. p. 3).

[FR16] Emmanuel Filiot et Pierre-Alain Reynier. « Transducers, Logic and Algebra for Functions of Finite Words ». In : *ACM SIGLOG News* 3.3 (août 2016), p. 4-19. issn : 2372-3491. doi : 10.1145/2984450.2984453 (cf. p. 4).

[Gir87] Jean-Yves Girard. « Linear logic ». In : *Theoretical Computer Science* 50.1 (jan. 1987), p. 1-101. issn : 0304-3975. doi : 10.1016/0304-3975(87)90045-4 (cf. p. 1, 6).

[Gir89] Jean-Yves Girard. « Towards a Geometry of Interaction ». In : *Categories in Computer Science and Logic*. Sous la dir. de John W. Gray et Andre Scedrov. T. 92. Contemporary Mathematics. Proceedings of a Summer Research Conference held June 14–20, 1987. Providence, RI : American Mathematical Society, 1989, p. 69-108. doi : 10.1090/conm/092/1003197 (cf. p. 7).

[Gir98] Jean-Yves Girard. « Light Linear Logic ». In : *Information and Computation* 143.2 (juin 1998), p. 175-204. issn : 0890-5401. doi : 10.1006/inco.1998.2700 (cf. p. 2).

[GKT01] Harold N. Gabow, Haim Kaplan et Robert Endre Tarjan. « Unique Maximum Matching Algorithms ». In : *Journal of Algorithms* (2001). doi : 10.1006/jagm.2001.1167 (cf. p. 6).

[GLN24] Paul Gallot, Nathan Lhote et Lê Thành Dũng Nguyễn. *Ambiguity/growth of tree automata/transducers made easy via MSO queries*. Slides available at https://nguyentito.eu/2024-02-labri.pdf. 2024 (cf. p. 5).

[GLS20] Paul Gallot, Aurélien Lemay et Sylvain Salvati. « Linear High-Order Deterministic Tree Transducers with Regular Look-Ahead ». In : *45th International Symposium on Mathematical Foundations of Computer Science, MFCS*

2020, August 24-28, 2020, Prague, Czech Republic. Sous la dir. de Javier Esparza et Daniel Kráľ. T. 170. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 38 :1-38 :13. doi : 10.4230/LIPIcs.MFCS.2020.38 (cf. p. 3).

[GMM23] Sam van Gool, Paul-André Melliès et Vincent Moreau. « Profinite lambda-terms and parametricity ». In : *Electronic Notes in Theoretical Informatics and Computer Science* Volume 3 – Proceedings of MFPS XXXIX (nov. 2023). doi : 10.46298/entics.12280 (cf. p. 3).

[Gou+13] Laurent Gourvès, Adria Lyra, Carlos A. Martinhon et Jérôme Monnot. « Complexity of trails, paths and circuits in arc-colored digraphs ». In : *Discrete Applied Mathematics* 161.6 (avr. 2013), p. 819-828. issn : 0166-218X. doi : 10.1016/j.dam.2012.10.025 (cf. p. 7).

[GS01] Alessio Guglielmi et Lutz Strassburger. « Non-commutativity and MELL in the Calculus of Structures ». en. In : *Computer Science Logic*. Sous la dir. de Gerhard Goos, Juris Hartmanis, Jan van Leeuwen et Laurent Fribourg. T. 2142. Berlin, Heidelberg : Springer Berlin Heidelberg, 2001, p. 54-68. isbn : 978-3-540-42554-0 978-3-540-44802-0. doi : 10.1007/3-540-44802-0_5 (cf. p. 7).

[GSS92] Jean-Yves Girard, Andre Scedrov et Philip J. Scott. « Bounded linear logic : a modular approach to polynomial-time computability ». In : *Theoretical Computer Science* 97.1 (avr. 1992), p. 1-66. issn : 0304-3975. doi : 10.1016/0304-3975(92)90386-T (cf. p. 2).

[Gue11] Stefano Guerrini. « A linear algorithm for MLL proof net correctness and sequentialization ». In : *Theoretical Computer Science* (2011). doi : 10.1016/j.tcs.2010.12.021 (cf. p. 6).

[Gug07] Alessio Guglielmi. « A system of interaction and structure ». en. In : *ACM Transactions on Computational Logic* 8.1 (jan. 2007). According to the author, the published version contains errors introduced by the editorial processing ; the authoritative version is https://arxiv.org/abs/cs/9910023. issn : 15293785. doi : 10.1145/1182613.1182614 (cf. p. 7).

[HHS19] Willem B. Heijltjes, Dominic J. D. Hughes et Lutz Strassburger. « Intuitionistic proofs without syntax ». In : *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*. IEEE, 2019, p. 1-13. doi : 10.1109/LICS.2019.8785827 (cf. p. 7).

[Hin03] Peter Hines. « A Categorical Framework For Finite State Machines ». In : *Mathematical Structures in Computer Science* 13.3 (2003), p. 451-480. doi : 10.1017/S0960129503003931 (cf. p. 6).

[Hin06] Peter Hines. *Temperley-Lieb Algebras as two-way automata*. Slides of a talk given at the QNET Workshop 2006. 2006. url : http://www.dcs.gla.ac.uk/~simon/qnet/talks/Hines.pdf (cf. p. 6).

[HK96] Gerd G. Hillebrand et Paris C. Kanellakis. « On the Expressive Power of Simply Typed and Let-Polymorphic Lambda Calculi ». In : *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 1996, p. 253-263. isbn : 978-0-8186-7463-1. doi : 10.1109/LICS.1996.561337 (cf. p. 2, 3).

[HKM96] Gerd G. Hillebrand, Paris C. Kanellakis et Harry G. Mairson. « Database Query Languages Embedded in the Typed Lambda Calculus ». In : *Information and Computation* 127.2 (juin 1996), p. 117-144. issn : 0890-5401. doi : 10.1006/inco.1996.0055 (cf. p. 2).

[Hug06] Dominic J.D. Hughes. « Proofs Without Syntax ». In : *Annals of Mathematics* 143.3 (nov. 2006), p. 1065-1076 (cf. p. 7).

[JM11] Paulin Jacobé de Naurois et Virgile Mogbil. « Correctness of Linear Logic Proof Structures is NL-Complete ». In : *Theoretical Computer Science* (2011). doi : 10.1016/j.tcs.2010.12.020 (cf. p. 6).

[KNP23] Sandra Kiefer, Lê Thành Dũng Nguyễn et Cécilia Pradic. *Refutations of pebble minimization via output languages*. Submitted to *Fundamenta Informaticae*. 2023. arXiv : 2301.09234 [cs.FL] (cf. p. 5).

[Lai13] James Laird. « Game semantics for a polymorphic programming language ». In : *Journal of the ACM* 60.4 (2013), 29 :1-29 :27. doi : 10.1145/2505986 (cf. p. 5).

[Lam58] Joachim Lambek. « The Mathematics of Sentence Structure ». In : *The American Mathematical Monthly* 65.3 (1958), p. 154-170. issn : 00029890, 19300972. url : http://www.jstor.org/stable/2310058 (cf. p. 2).

[Lho20] Nathan Lhote. « Pebble Minimization of Polyregular Functions ». In : *LICS '20 : 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*. Sous la dir. d'Holger Hermanns, Lijun Zhang, Naoki Kobayashi et Dale Miller. ACM, 2020, p. 703-712. doi : 10.1145/3373718.3394804 (cf. p. 4).

[Mel17] Paul-André Melliès. « Higher-order parity automata ». en. In : *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. Reykjavik, Iceland : IEEE, juin 2017, p. 1-12. isbn : 978-1-5090-3018-7. doi : 10.1109/LICS.2017.8005077 (cf. p. 3).

[MN23] Vincent Moreau et Lê Thành Dũng Nguyễn. *Syntactically and semantically regular languages of lambda-terms coincide through logical relations*. To appear in the proceedings of CSL 2024. 2023. arXiv : 2308.00198 (cf. p. 3).

[MP19] Anca Muscholl et Gabriele Puppis. « The Many Facets of String Transducers ». In : *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*. Sous la dir. de Rolf Niedermeier et Christophe Paul. T. 126. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 2 :1-2 :21. isbn : 978-3-95977-100-9. doi : 10.4230/LIPIcs.STACS.2019.2 (cf. p. 4).

[Ngu19a] Lê Thành Dũng Nguyễn. « Around finite second-order coherence spaces ». In : *CoRR* abs/1902.00196 (2019). arXiv : 1902.00196 (cf. p. 5).

[Ngu19b] Lê Thành Dũng Nguyễn. « Constrained path-finding and structure from acyclicity ». In : *CoRR* abs/1901.07028 (2019). arXiv : 1901.07028 (cf. p. 7).

[Ngu19c] Lê Thành Dũng Nguyễn. « On the Elementary Affine Lambda-Calculus with and Without Fixed Points ». In : *Electronic Proceedings in Theoretical Computer Science* 298 (août 2019). In Proceedings DICE-FOPARA 2019, p. 15-29. ISSN : 2075-2180. DOI : 10.4204/EPTCS.298.2 (cf. p. 5).

[Ngu20] Lê Thành Dũng Nguyễn. « Unique perfect matchings, forbidden transitions and proof nets for linear logic with Mix ». In : *Logical Methods in Computer Science* 16.27 (fév. 2020). DOI : 10.23638/LMCS-16(1:27)2020 (cf. p. 6, 7).

[Ngu21] Lê Thành Dũng Nguyễn. « Implicit automata in linear logic and categorical transducer theory ». Thèse de doct. Université Paris XIII (Sorbonne Paris Nord), déc. 2021. URL : https://hal.science/tel-04132636 (cf. p. 2-4).

[Ngu23] Lê Thành Dũng Nguyễn. *Simply typed convertibility is TOWER-complete even for safe lambda-terms*. Accepted subject to minor revisions in *Logical Methods in Computer Science*. 2023. arXiv : 2305.12601 [cs.LO] (cf. p. 6).

[NNP21] Lê Thành Dũng Nguyễn, Camille Noûs et Cécilia Pradic. « Comparison-Free Polyregular Functions ». In : *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*. Sous la dir. de Nikhil Bansal, Emanuela Merelli et James Worrell. T. 198. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany : Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 139 :1-139 :20. ISBN : 978-3-95977-195-5. DOI : 10.4230/LIPIcs.ICALP.2021.139 (cf. p. 4).

[NNP23] Lê Thành Dũng Nguyễn, Camille Noûs et Cécilia Pradic. *Two-way automata and transducers with planar behaviours are aperiodic*. Submitted to *Discrete Mathematics and Theoretical Computer Science*. 2023. arXiv : 2307.11057 [cs.FL] (cf. p. 6, 7).

[NP19] Lê Thành Dũng Nguyễn et Cécilia Pradic. « From normal functors to logarithmic space queries ». In : *46th International Colloquium on Automata, Languages and Programming (ICALP 2019)*. Sous la dir. de Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini et Stefano Leonardi. T. 132. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 123 :1-123 :15. ISBN : 978-3-95977-109-2. DOI : 10.4230/LIPIcs.ICALP.2019.123 (cf. p. 5).

[NP20] Lê Thành Dũng Nguyễn et Cécilia Pradic. « Implicit automata in typed λ-calculi I : aperiodicity in a non-commutative logic ». In : *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*. Sous la dir. d'Artur Czumaj, Anuj Dawar et Emanuela Merelli. T. 168. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 135 :1-135 :20. DOI : 10.4230/LIPIcs.ICALP.2020.135 (cf. p. 2, 3, 6).

[NS18] Lê Thành Dũng Nguyễn et Thomas Seiller. « Coherent Interaction Graphs ». In : *Proceedings Joint International Workshop on Linearity & Trends in Linear Logic and Applications, Linearity-TLLA@FLoC 2018, Oxford, UK, 7-8 July 2018*. Sous la dir. de Thomas Ehrhard, Maribel Fernández, Valeria de Paiva et Lorenzo Tortora de Falco. T. 292. EPTCS. 2018, p. 104-117. DOI : 10.4204/EPTCS.292.6 (cf. p. 7).

[NS23] Lê Thành Dũng Nguyễn et Lutz Strassburger. « A System of Interaction and Structure III : The Complexity of BV and Pomset Logic ». In : *Logical Methods in Computer Science* Volume 19, Issue 4 (déc. 2023). DOI : 10.46298/lmcs-19(4:25)2023 (cf. p. 7).

[NV24] Lê Thành Dũng Nguyễn et Gabriele Vanoni. *Invisible pebbles and the geometry of higher-order affine transducers*. Presented at the 15th Workshop on Games for Logic and Programming Languages. 2024. URL : https://nguyentito.eu/tmp/galop24.pdf (cf. p. 3, 7).

[Péc20] Romain Péchoux. « Implicit Computational Complexity : past and future (Complexité implicite : bilan et perspectives) ». Habilitation à diriger des recherches. Université de Lorraine, oct. 2020. URL : https://hal.univ-lorraine.fr/tel-02978986 (cf. p. 2).

[Ret03] Christian Retoré. « Handsome proof-nets : perfect matchings and cographs ». In : *Theoretical Computer Science*. Linear Logic 294.3 (fév. 2003), p. 473-488. ISSN : 0304-3975. DOI : 10.1016/S0304-3975(01)00175-X (cf. p. 6, 7).

[Ret97] Christian Retoré. « Pomset logic : A non-commutative extension of classical linear logic ». en. In : *Typed Lambda Calculi and Applications*. T. 1210. Berlin, Heidelberg : Springer Berlin Heidelberg, 1997, p. 300-318. ISBN : 978-3-540-62688-6 978-3-540-68438-1. DOI : 10.1007/3-540-62688-3_43 (cf. p. 7).

[Sal09] Sylvain Salvati. « Recognizability in the Simply Typed Lambda-Calculus ». In : *Logic, Language, Information and Computation, 16th International Workshop, WoLLIC 2009, Tokyo, Japan, June 21-24, 2009. Proceedings*. Sous la dir. d'Hiroakira Ono, Makoto Kanazawa et Ruy J. G. B. de Queiroz. T. 5514. Lecture Notes in Computer Science. Springer, 2009, p. 48-60. DOI : 10.1007/978-3-642-02261-6_5 (cf. p. 3).

[Sal15] Sylvain Salvati. « Lambda-calculus and formal language theory ». Habilitation à diriger des recherches. Université de Bordeaux, déc. 2015. URL : https://tel.archives-ouvertes.fr/tel-01253426 (cf. p. 3).

[Sch16] Sylvain Schmitz. « Complexity Hierarchies beyond Elementary ». In : *ACM Transactions on Computation Theory* 8.1 (2016), 3 :1-3 :36. DOI : 10.1145/2858784 (cf. p. 5).

[Sei12] Thomas Seiller. « Logique dans le Facteur Hyperfini : Géométrie de l'Interaction et Complexité ». Thèse de doct. Aix-Marseille Université, nov. 2012. URL : https://tel.archives-ouvertes.fr/tel-00768403 (cf. p. 7).

[Sta79] Richard Statman. « The typed λ-calculus is not elementary recursive ». In : *Theoretical Computer Science* 9.1 (juill. 1979), p. 73-81. ISSN : 0304-3975. DOI : 10.1016/0304-3975(79)90007-0 (cf. p. 5, 6).

[ZG15] Noam Zeilberger et Alain Giorgetti. « A correspondence between rooted planar maps and normal planar lambda terms ». In : *Logical Methods in Computer Science* 11.3 (sept. 2015). ISSN : 18605974. DOI : 10.2168/LMCS-11(3:22)2015 (cf. p. 2).