PROJET DE RECHERCHE : CALCULABILITÉ PAR ÉTATS FINIS, OPTIMISATION ET PHÉNOMÈNES D'ORDRE SUPÉRIEUR

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

Il est recommandé de lire préalablement les sections 1 à 4 du rapport sur les travaux antérieurs, auquel le présent document fait référence par le sigle RTA.

Au commencement étaient divers modèles de calcul : fonctions récursives partielles, λ-calcul non typé, machines de Turing, etc. Se révélant *équivalents* en pouvoir calculatoire, autrement dit : définissant tous la même classe de fonctions, ils inspirèrent la *thèse de Church–Turing*, selon laquelle cette classe reflète fidèlement la notion informelle de *fonction calculable*. Si cette notion absolue de calculabilité est devenue une banalité avec l'essor de l'informatique (la « Turing-complétude accidentelle » est un phénomène courant), il ne serait pas absurde d'imaginer un autre univers : un scénario de science-fiction où le domaine du calculable s'étendrait au fur et à mesure de la découverte de nouveaux principes (physiques[1] ou algorithmiques) sans atteindre de point final.

C'est en fait ce qui se passe en *théorie des transducteurs*. Précisons le propos, afin qu'on ne se dise pas « après tout, une machine de Turing est une sorte d'automate… » : je parle ici uniquement de machines qu'on peut raisonnablement considérer comme étant « à états finis ». En particulier, tous les modèles de transducteurs dont il sera question ici donnent, quand on les restreint à produire un booléen, la classe des *langages rationnels*. En effet, s'agissant des fonctions des mots vers des booléens – c'est-à-dire des « langages » ou « problèmes de décision » – de nombreuses définitions (automates déterministes ou non-déterministes, reconnaissance par semigroupes, …) convergent vers les langages rationnels, qu'on considère donc comme une notion robuste et canonique de *calculabilité par états finis*. Par contre, lorsqu'on considère des *fonctions des mots vers les mots*, on dispose à présent de divers modèles de transducteurs plus ou moins puissants, mais il n'y a pas de raison de croire que l'histoire s'arrête aux modèles déjà connus.

Mon projet est de **mieux cerner ces fonctions calculables par états finis** : à défaut de pouvoir les définir précisément pour l'instant, que peut-on dire des propriétés qu'elles devraient vérifier ?

1. Polyrégularité = calculabilité *efficace* par états finis ?

Une première piste, suggérée par l'expérience acquise durant ces dernières années au sujet des fonctions (poly)régulières (voir RTA§4), est la suivante.

Thèse conjecturelle 1. Les fonctions entre mots calculables par états finis et à *croissance polynomiale* (c'est-à-dire $|f(w)| = |w|^{O(1)}$) devraient être exactement les fonctions *polyrégulières*.

Ce n'est pas une conjecture bien posée, puisqu'on ne sait pas ce que signifie « calculable par états finis » ; cette « thèse » (mot employé avec le même sens que dans « thèse de Church–Turing ») est justement censée nous éclairer là-dessus. Mais on peut en tirer des questions plus concrètes, *relatives* à des classes de fonctions $\mathcal{C}$ considérées comme faisant partie de « l'univers des états finis ».

Conjecture 2 (dépendant de $\mathcal{C}$)**.** *Toute fonction dans $\mathcal{C}$ à croissance polynomiale est polyrégulière.*

On peut envisager un intérêt pratique : mettons que $\mathcal{C}$ soit la classe des fonctions définissables dans un langage de programmation assez expressif, laissant beaucoup de liberté pour décrire des transformations de données, mais en restant « à états finis » pour que la conjecture soit plausible.

[1] Concernant les fonctions *efficacement* calculables, les ordinateurs quantiques et la conjecture $\mathsf{P} \neq \mathsf{BQP}$ remettent en cause la « thèse de Church–Turing étendue » affirmant que tous les modèles de calculs à « modèle de coût raisonnable » définissent la même notion de calculabilité en temps polynomial.

Même si $\mathcal{C}$ contient des fonctions à croissance exponentielle (ou pire), on voudra souvent écrire des programmes « raisonnables » qui ne produisent pas des sorties trop grandes. Si la conjecture s'avérait vraie pour la classe $\mathcal{C}$, alors ces programmes raisonnables pourraient être traduits vers des transducteurs à jetons (qui servent à définir les fonctions polyrégulières). Or on connaît des algorithmes d'évaluation efficaces pour les transducteurs à jetons [Boj18, Part III]. Cela donnerait donc une forme de *compilation optimisante*, à condition qu'on dispose d'un programme qui réalise cette traduction. C'est pourquoi on est amené à raffiner l'énoncé :

Conjecture 3 (Version effective de la Conjecture 2, dépendant de $\mathcal{C}$)**.** *Il existe un algorithme prenant en entrée une fonction $f \in \mathcal{C}$ décrite dans un formalisme caractérisant la classe $\mathcal{C}$ (langage de programmation ou modèle de machines ou système logique ou …) qui détermine si f est à croissance polynomiale et, si elle l'est, renvoie une description de f par un transducteur à jetons qui la calcule.*

Notons un parallèle avec la complexité implicite (cf. RTA§2) : les recherches dans ce domaine ont inspiré des optimisations dans des compilateurs [Rub17] ainsi que des analyses statiques de complexité de programmes (voir [Péc20, §1.2.6] pour un survey ou [Aub+22 ; Aub+23] pour un travail récent). Ces analyses sont correctes mais incomplètes, puisque savoir si un programme tourne en temps polynomial (par exemple) est en général indécidable. Par contre, dans un cadre « à états finis », des problèmes analogues admettent parfois des procédures de décision correctes et complètes à la fois ; c'est souvent le cas pour l'appartenance d'un langage rationnel donné à une sous-classe fixée provenant de la théorie algébrique des automates. À ce propos, signalons que quelques-unes de ces sous-classes – notamment la classe des langages sans étoile évoquée dans RTA§2 – ont récemment été mises en lien avec des problématiques d'optimisation bas niveau (instructions SIMD) pour le traitement de fichiers texte [Soy23 ; GMP24].

Quant à l'intérêt *théorique* de la Thèse conjecturelle 1 (et donc des conjectures qui en découlent), il est clair : elle signifierait que la polyrégularité est à la notion insaisissable de calculabilité par états finis ce que la complexité en temps polynomial est à la Turing-calculabilité.

Dans mon manuscrit de thèse, j'énonce l'instance de la Conjecture 2 obtenue en prenant comme classe $\mathcal{C}$ celle des **fonctions définissables en λ-calcul simplement typé** [Ngu21, Conj. 1.4.14] – en fait, la Conjecture 3 semble tout autant plausible dans ce cas. Justifier la pertinence de ce choix nécessite de rappeler un peu de contexte ; c'est l'objet des sections suivantes.

2. Une classe importante de fonctions à états finis entre arbres

Des réponses positives pour certaines instances des Conjectures 2 et 3 seraient d'autant plus intéressantes que les classes $\mathcal{C}$ considérées sont grandes. Or la classe de fonctions que j'appelle $\mathcal{E}$ dans [Ngu21, §1.4.1] (et que j'utilise dans [KNP23, §4]), introduite dans les travaux d'Engelfriet et al., inclut à ma connaissance toutes les autres sérieusement étudiées dans la littérature sur les transducteurs. Les fonctions $f \in \mathcal{E}$ peuvent être caractérisées comme les composées $f = f_1 \circ \cdots \circ f_k$ ($k \in \mathbb{N}$) de celles calculées par des transducteurs relativement simples [EV86 ; EM03].

Ces transducteurs calculent des fonctions entre *arbres*, et non seulement entre mots (les mots pouvant être représentés comme arbres unaires). Cela fait une vraie différence dans le cas de ces machines trop faibles pour parser la sérialisation d'un arbre. La théorie des automates d'arbres a ainsi été appliquée au traitement de documents à structure hiérarchique (un exemple classique : valider un document XML ou JSON par rapport à un schéma) tandis que les transducteurs d'arbres servent à calculer des transformations de tels documents (c'est pourquoi, typiquement, XSLT est cité comme motivation de l'article introduisant les transducteurs d'arbres à jetons [MSV03]).

En plus de leur simplicité, les « macro tree transducers » de [EV86] et transducteurs cheminants (« tree-walking ») de [EM03][2] **reflètent les langages rationnels d'arbres** – de même, tous les transducteurs dont il avait été question précédemment reflétaient les langages rationnels de mots.

Définition 4. Une fonction $f : \mathrm{Tree}(\Sigma) \to \mathrm{Tree}(\Gamma)$ *reflète les langages rationnels d'arbres* lorsque pour tout $L \subseteq \mathrm{Tree}(\Gamma)$ rationnel, son image réciproque $f^{-1}(L) \subseteq \mathrm{Tree}(\Sigma)$ est également rationnelle.

[2] Appelés « 0-pebble transducers » dans [EM03], mais la terminologie « tree-walking » est employée par exemple par [AU71] dans le cas de fonctions des arbres vers les mots, et par [EIM21] dans le cas général.

Pour ces raisons, on n'hésite pas à considérer les transducteurs cheminants et « macro » comme relevant de la calculabilité par états finis. Par conséquent, la classe $\mathcal{E}$ devrait aussi en relever, selon le principe suivant : **si deux fonctions sont calculables par états finis, leur composée l'est aussi**.

De plus, cette classe de fonctions vérifie une propriété proche de la Conjecture 3, mais pour la croissance *linéaire* plutôt que polynomiale.

Théorème 5 ([EIM21])**.** *Les fonctions à* croissance linéaire ($|f(w)| = O(|w|)$) *dans $\mathcal{E}$ sont exactement les fonctions* régulières *entre arbres. De plus, on a un algorithme qui décide si une fonction dans $\mathcal{E}$ est à croissance linéaire et, le cas échéant, en renvoie une représentation en tant que fonction régulière.*

Il pourrait donc être raisonnable à court terme de s'attaquer à l'instance de la Conjecture 3 elle-même pour $\mathcal{E}$. Un détail : la conjecture parle de fonctions entre mots, et non entre arbres ; mais je m'attends à voir une bonne notion de fonction polyrégulière entre arbres émerger de l'équipe d'automates de Varsovie dans le futur proche (un candidat naturel serait les MSO-interprétations de [BKL19], qui caractérisent les polyrégulières sur les mots et s'étendent facilement aux arbres).

3. Transducteurs d'ordre supérieur et λ-calcul

Deux autres modèles de machines caractérisent directement $\mathcal{E}$ sans parler de composition : les transducteurs à piles imbriquées [EV86 ; EV88] et les « high level tree transducers » [EV88]. Ces derniers stockent en mémoire des *fonctions d'ordre supérieur*. Les transducteurs « macro » utilisent en fait des contextes d'arbres qu'on déjà peut voir comme des fonctions « d'ordre 1 » très restreintes : le contexte $a(a(\square_1, b), \square_2)$ avec deux trous $\square_1, \square_2$ représente $(x, y) \mapsto a(a(x, b), y)$. Et lorsqu'on compose $k \geqslant 2$ transducteurs « macro », la fonction obtenue peut être calculée directement par un transducteur stockant des fonctions « d'ordre k », c'est-à-dire prenant des arguments eux-même d'ordre au plus $k - 1$: ce sont des fonctions sur les fonctions, manipulées comme données. Ainsi, la composition nous amène sur le terrain de la *programmation fonctionnelle*, et donc du *λ-calcul*.

Cela rejoint donc les travaux que j'ai menés durant mon doctorat avec Cécilia Pradic sur les « automates implicites » [NP20 ; Ngu21], et plus récemment avec Gabriele Vanoni [NV24] sur les « λ-transducteurs » (suivant l'approche de [GLS20]) – voir RTA§2. Les « high level transducers » de [EV88], qui caractérisent $\mathcal{E}$, peuvent être vus comme des λ-transducteurs utilisant un fragment légèrement restreint du *λ-calcul simplement typé* (STλC) – j'en reparlerai plus loin.

À ce stade, il est naturel de se pencher sur les « λ-transducteurs simplement typés » sans cette légère restriction. Ils sont déjà connus sous le nom de *transducteurs d'ordre supérieur*, mais semblent étonnamment n'apparaître quasiment pas dans la littérature.[3] En fait, on voit facilement qu'ils sont capables de calculer précisément les fonctions entre arbres définissables à l'intérieur du λ-calcul simplement typé (avec des codages de Church[4]) – c'est le point de vue « automates implicites », qui s'avère ici équivalent aux « λ-transducteurs ». Ce qu'on sait de la STλC-définissabilité se résume à quelques résultats classiques d'inexpressibilité – par exemple, sur des codages d'entiers, on ne peut pas exprimer la soustraction[5] – qu'on peut en fait retrouver comme corollaires triviaux du théorème de Hillebrand et Kanellakis [HK96] (cf. RTA§2).

Ces fonctions STλC-définissables doivent-elles toutes être considérées comme calculables par états finis ? Une réponse positive est fortement suggérée par :

- leur proximité avec $\mathcal{E}$ (qui, nous l'avons vu, est clairement « à états finis ») ;
- le fait qu'elles reflètent les langages rationnels (autre conséquence immédiate de [HK96]) ;
- l'usage habituel de sémantiques dénotationnelles finitaires (mentionnées en RTA§3) pour des analyses statiques de λ-termes simplement typés, de façon analogue à ce qui se passe en théorie des automates – voir [Sal15] ou encore la section « related work » de [MN23].

[3] Avant 2020, la principale mention semble être l'exposé d'Inaba en 2013 que je cite à la fin de la prochaine section. Citons aussi les travaux [KTU10 ; OR11] sur la vérification formelle de programmes fonctionnels, qui utilisent des machines avec quelques points communs avec les transducteurs d'ordre supérieur.

[4] Avec la même convention d'entrée-sortie que dans [HK96], autorisant une substitution dans le type d'entrée. Sans cette substitution, on obtient une classe bien plus petite qui admet des caractérisations alternatives [Zai91 ; Lei93].

[5] Résultat mentionné notamment dans [FLO83] où il est attribué à Richard Statman.

Dans ce cas, la classe des fonctions STλC-définissables engloberait la totalité du territoire connu au sein des fonctions calculables par états finis, ce qui justifierait de s'attaquer aux Conjectures 2 et 3 pour cette classe ; voire, plus généralement, d'étudier davantage ces fonctions.

4. Safe vs unsafe : un saut qualitatif potentiel

On peut caractériser la classe $\mathcal{E}$ dans un style « automates implicites » au moyen du λ-calcul *safe*[6] de [BO09]. En effet, les « high level transducers » de [EV88], reposent sur la notion de « derived type » [Dam82], dont une relecture basée sur la « safety » a été proposée ultérieurement dans le cadre de travaux sur les *schémas récursifs d'ordre supérieur* [KNU02 ; SW16].

L'idée est la suivante : on a deux formalismes naturels pour décrire des arbres infinis, l'un étant basé sur le λ-calcul simplement typé (étendu avec de la récursivité, d'où le nom de « schémas récursifs ») et l'autre étant un modèle d'automates. Ils correspondent presque... mais la version automates est un peu moins expressive. Pour remédier à ce décalage, on peut soit imposer la safety côté λ-calcul pour le brider, soit, dans l'autre sens, rendre les automates plus puissants [Hag+17].

Le modèle d'automates « naturel » susmentionné est celui des automates à piles imbriquées, et son extension dans [Hag+17] consiste à lui rajouter une opération de « collapse ». Rappelons maintenant que les transducteurs à piles imbriquées caractérisent $\mathcal{E}$ [EV86]. En vertu des analogies fortes qu'on voit apparaître entre transducteurs calculant une fonction entre structures finies, et schémas engendrant une structure infinie, on devrait avoir :

Affirmation 6 ([Ngu21, Conjecture 1.4.3])**.** *Il existe un modèle de « transducteurs à piles imbriquées avec* collapse *» qui sait calculer précisément les fonctions STλC-définissables.*

Je me suis déjà convaincu que je savais prouver cette affirmation, en réunissant des ingrédients issus de [Hag+17 ; SW16 ; Plo22] ; mais reste à écrire la démonstration rigoureusement... Au fond, l'intérêt de cet énoncé est surtout psychologique : il fournit un argument supplémentaire pour convaincre la communauté de théorie des automates que la STλC-définissabilité relève bien de la calculabilité par états finis, puisqu'elle peut être décrite par un modèle de transducteurs.

Un fait qui me semble plus intrinsèquement intéressant est que les schémas safe sont strictement moins expressifs que les schémas unsafe [Par20]. Par analogie, on soupçonne donc que :

Conjecture 7. *Il existe une fonction STλC-définissable entre arbres qui n'est pas dans $\mathcal{E}$.*

Pour attaquer cette conjecture, le point de départ évident est d'étudier la preuve de Parys [Par20] pour les schémas. Le souci, c'est que c'est l'une de ces démonstrations très longues et techniques que quasiment personne ne comprend. À mon avis, c'est à cause de l'utilisation dans cette preuve d'automates à piles imbriquées, dont la combinatoire est complexe.

Dans le cas des schémas unsafe, certains résultats démontrés dans un premier temps à l'aide des automates à collapse ont été reprouvés de façon bien plus simple techniquement et éclairante conceptuellement par des méthodes sémantiques.[7] Idéalement, on souhaiterait obtenir une démonstration sémantique du théorème de Parys, éventuellement en reformulant sémantiquement les idées combinatoires de [Par20]. Un obstacle à cela est que la safety n'est pas très naturelle d'un point de vue sémantique – en fait, elle a été inventée pour les besoins de la théorie des schémas récursifs, et n'apparaît pas ailleurs. Mais de fortes ressemblances entre la safety et des conditions de « stratification » utilisées pour la complexité implicite en logique linéaire m'amènent à croire (et je pense savoir le prouver dans le cas des fonctions entre arbres finis) que :

Affirmation 8. *La* logique linéaire élémentaire[8] propositionnelle [DJ03] *définit exactement les fonctions entre arbres finis dans $\mathcal{E}$, et les schémas récursifs linéaires élémentaires engendrent exactement les mêmes arbres potentiellement infinis que les schémas safe.*

---

[6] J'ai d'ailleurs résolu une question ouverte sur ce λ-calcul « sûr » grâce à des transducteurs [Ngu23], cf. RTA§5.

[7] Citons deux exemples : la décidabilité du problème de sélection MSO [Gre16, Chapter 10], et un lemme de pompage appliqué à séparer les niveaux de la hiérarchie des schémas unsafe [Kob13]. Ce dernier article utilise un système de types intersection, mais cela revient au même qu'une sémantique dénotationnelle, cf. [Ron18].

[8] J'ai travaillé sur la logique linéaire élémentaire *du 2nd ordre* – et non propositionnelle – dans les articles [Ngu19 ; NP19].

Comme on comprend bien les sémantiques dénotationnelles de la logique linéaire élémentaire propositionnelle [Lau09], cela pourrait ouvrir la voie à l'extension des méthodes sémantiques aux schémas safe ainsi qu'aux fonctions dans $\mathcal{E}$, par exemple pour avancer sur la Conjecture 7.

Inversement, un outil théorique dont on dispose dans le cas safe mais pas dans le cas unsafe est la décomposition en suite de transducteurs plus simples [EV86 ; EM03] – c'est la première caractérisation de $\mathcal{E}$ qui a été présentée ici. (Une décomposition analogue existe pour les schémas safe [Cau02 ; CW03].) C'est notamment cette caractérisation de $\mathcal{E}$ qui est utilisée pour démontrer le Théorème 5 par récurrence sur la longueur de la décomposition. Il n'est pas clair que les fonctions STλC-définissables admettent un théorème semblable de décomposition : la question a été posée pour les transducteurs d'ordre supérieur dans un exposé de Kazuhiro Inaba à Dagstuhl il y a une décennie [MS13, §3.7][9] et n'a toujours pas reçu de réponse à ce jour. Cela signifie qu'on ne peut pas facilement adapter la preuve du Théorème 5 pour l'étendre de $\mathcal{E}$ aux fonctions STλC-définissables ; et, a fortiori, que la Conjecture 3 pour ces dernières s'annonce difficile.

5. Peut-on avoir une thèse de Church–Turing à états finis ?

Revenons désormais à notre motivation initiale : mieux comprendre les fonctions entre mots. Nous avons à plusieurs reprises invoqué la propriété de réflexion des langages rationnels, comme indice heuristique qu'une classe de fonctions est bien incluse dans l'univers des états finis. Il s'agit en tout cas d'une condition nécessaire ; toutefois, elle est loin d'être suffisante. Une illustration en est donnée par la famille suivante d'exemples, tirée de mon article avec Mikołaj Bojańczyk.

Exemple 9 ([BN23, Ex. 2.12])**.** Soit $g\colon \mathbb{N} \to \mathbb{N}$ une fonction croissante dont l'image est incluse dans $\{1!, 2!, 3!, \dots\}$ où « ! » désigne la factorielle. Alors $f\colon \{a\}^* \to \{a\}^*$ définie par $f(a^n) = a^{g(n)}$ reflète les langages rationnels. En particulier, pour $h\colon \mathbb{N} \to \{0,1\}$, on en déduit la réflexion des langages rationnels pour $f\colon a^n \mapsto a^{(\mathrm{Card}\{m \mid m! \leqslant n \text{ et } h(m)=1\})!}$; de plus, $|f(w)| \leqslant |w|$. Or f est susceptible de partager les « pathologies » de h (par exemple, si h n'est pas Turing-calculable, alors f non plus).

Si l'on croit à la Thèse conjecturelle 1, la question est alors : quelles conditions supplémentaires imposer pour exclure de telles pathologies et forcer f à être régulière[10] ? L'approche proposée pour l'instant a été de considérer des conditions du type « $f \in \mathcal{C}$ pour $\mathcal{C}$ une classe de fonctions considérée "à états finis" ». Mais on pourrait aussi rechercher une caractérisation axiomatique des fonctions (poly)régulières qui ne fasse pas intervenir l'existence d'une description « syntaxique » de la fonction (que ce soit par une machine, un programme ou une formule logique). Idéalement, l'un des axiomes serait la croissance linéaire/polynomiale, et les autres lui seraient en quelque sorte orthogonaux ; ce qui rejoindrait une problématique plus générale :

Question 10. Peut-on trouver des conditions que *toute* fonction calculable par états finis vérifie nécessairement, en plus de la réflexion des langages rationnels et la Turing-calculabilité ?

Autrement dit, il s'agirait de trouver des « majorants » à l'univers des états finis. Quant aux « minorants », si le meilleur dont on dispose est la STλC-définissabilité, on peut explorer d'autres façons de décrire des fonctions entre mots ou entre arbres, par exemple par spécification logique (théorie des modèles) : c'est ce que font les MSO interprétations de [BKL19], qui caractérisent les polyrégulières. Elles sont moins générales que les « MSO set interpretations » [CL07] que j'étudie dans un travail en cours [GLN24]. Nathan Lhote cherche en ce moment à démontrer que les MSO set interpretations entre mots reflètent les langages rationnels – propriété qui était déjà difficile à établir pour les interprétations de [BKL19], alors qu'elle est facile pour de nombreux modèles de machines et pour les fonctions STλC-définissables.[11] Ces difficultés techniques suggèrent que quelque chose reste à clarifier là-dedans.

[9] Diapositives : `https://www.kmonos.net/pub/Presen/dagstuhl.pdf`

[10] La Thèse conjecturelle 1 parle de fonctions polyrégulières, mais une fonction polyrégulière à croissance linéaire est régulière – c'est un corollaire du Théorème 5.

[11] Le théorème de Hillebrand et Kanellakis [HK96] est en effet techniquement facile a posteriori, une fois qu'on connaît le bon outil, ce qui ne l'empêche pas d'être conceptuellement profond. (J. H. C. Whitehead disait : « It is the snobbishness of the young to suppose that a theorem is trivial because the proof is trivial. »)

Enfin, comme les MSO set interpretations sont à croissance exponentielle, leur clôture par composition contient des fonctions à croissance hyperexponentielle. C'est aussi le cas de $\mathcal{E}$ et de la classe des fonctions STλC-définissables ; il serait donc intéressant de comparer ces classes.

Intégration

Étant donné le positionnement de mon projet à l'interface de deux communautés distinctes (automates et preuves-programmes), le fait d'avoir des interlocuteurices pour chacune des deux constitue mon premier critère de choix pour mes propositions d'affectation.

LIP (ÉNS Lyon). Je suis en post-doctorat depuis septembre 2022 au sein de l'équipe *Plume*, qui est peut-être l'un des endroits au monde où les liens entre automates et logique linéaire sont les plus étudiés : je pense notamment aux thèses de ma coautrice principale Cécilia Pradic [Pra20] (co-encadrée[12] par **Colin Riba**) et de Laureline Pinault [Pin21] (encadrée par **Damien Pous** et **Denis Kuperberg**). Certains travaux d'**Amina Doumane** s'inscrivent dans cette lignée, et d'autres dans la théorie des transducteurs. **Olivier Laurent**, **Michele Pagani** et **Paolo Pistone** sont des spécialistes en λ-calcul (linéaire) et sémantique, avec qui je peux interagir aussi bien au sujet du projet présenté ici que du prolongement naturel de mes travaux sur la combinatoire des preuves (cf. RTA§6) ; Olivier et moi enseignons ensemble un cours de théorie des catégories dans le M2 informatique de l'ÉNS Lyon. **Matteo Mio** et **Valeria Vignudelli** travaillent entre autres sur la théorie catégorique des automates (cf. RTA§3) ; dans cette thématique, Damien et Valeria donnent un cours de M2 de coalgèbre qui s'appuie sur des prérequis introduits dans mon cours de catégories.

LIS (Aix-Marseille). Dans l'idéal, je viserais un rattachement simultané aux deux équipes *LSC*[13] et *Move*, mon projet s'inscrivant à la fois dans les thématiques de l'une et de l'autre. Côté *Move*, **Pierre-Alain Reynier**, **Jean-Marc Talbot** et **Nathan Lhote** sont des théoriciens des transducteurs ; j'ai déjà une collaboration en cours avec ce dernier [GLN24] (cf. RTA§4), qui utilise les automates pondérés dont **Benjamin Monmege** est spécialiste. Au sein de *LSC*, **Pierre Clairambault** est un expert sur de nombreux outils que j'utilise ou prévois d'utiliser dans mes recherches (géométrie de l'interaction, schémas récursifs d'ordre supérieur, ...) ; je pourrais également bénéficier de la collaboration de **Raphaëlle Crubillé** en sémantique catégorique. **Luigi Santocanale** est responsable local du projet ANR LambdaComb, sur lequel j'ai travaillé comme post-doctorant pendant 6 mois (avec Noam Zeilberger à Saclay) et dont les thèmatiques rejoignent mes travaux de combinatoire des preuves. Enfin, le LIS se situe à côté de l'Institut de Mathématiques de Marseille, dont l'équipe *Logique de la programmation* constitue l'un des pôles historiques de la logique linéaire.

IRIF (Université Paris Cité). Je souhaiterais m'intégrer à la fois aux trois équipes thématiques *Algèbre et calcul*, *Preuves et programmes* et *Automates et applications*. C'est déjà le cas pour certains personnels CNRS, par exemple **Paul-André Melliès** avec qui je partage de nombreux intérêts de recherche ; il encadre avec **Sam van Gool** la thèse de Vincent Moreau, qui a co-écrit avec moi une publication [MN23]. L'IRIF est connu pour être un des centres majeurs de la correspondance preuves-programmes en France (en particulier de la logique linéaire : **Thomas Ehrhard**, **Delia Kesner**, **Alexis Saurin**, ...). Concernant les automates, je pourrais interagir au sujet de leur théorie catégorique avec **Thomas Colcombet** et **Daniela Petrişan**, dont l'article commun [CP20] a nourri ma thèse (RTA§3), mais aussi au sujet des transducteurs avec **Sarah Winter** ou encore **Olivier Carton** (qui a encadré une thèse liée à mes travaux [Dou23], cf. RTA§4).

[12] Cette thèse en cotutelle avec l'Université de Varsovie a également été dirigée par Henryk Michalewski.

[13] Un *fork* récent de l'équipe LIRICA.

Références

[AU71] Alfred V. Aho et Jeffrey D. Ullman. « Translations on a Context-Free Grammar ». In : *Information and Control* 19.5 (1971). Journal version of a STOC 1969 paper, p. 439-475. doi : 10.1016/S0019-9958(71)90706-6 (cf. p. 2).

[Aub+22] Clément Aubert, Thomas Rubiano, Neea Rusch et Thomas Seiller. « mwp-Analysis Improvement and Implementation : Realizing Implicit Computational Complexity ». In : *7th International Conference on Formal Structures for Computation and Deduction, FSCD 2022, August 2-5, 2022, Haifa, Israel*. Sous la dir. d'Amy P. Felty. T. 228. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 26 :1-26 :23. doi : 10.4230/LIPICS.FSCD.2022.26 (cf. p. 2).

[Aub+23] Clément Aubert, Thomas Rubiano, Neea Rusch et Thomas Seiller. « pymwp : A Static Analyzer Determining Polynomial Growth Bounds ». In : *Automated Technology for Verification and Analysis - 21st International Symposium, ATVA 2023, Singapore, October 24-27, 2023, Proceedings, Part II*. Sous la dir. de Étienne André et Jun Sun. T. 14216. Lecture Notes in Computer Science. Springer, 2023, p. 263-275. doi : 10.1007/978-3-031-45332-8_14 (cf. p. 2).

[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. 3, 5).

[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. 5).

[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. 4).

[Boj18] Mikołaj Bojańczyk. « Polyregular Functions ». In : *CoRR* abs/1810.08760 (oct. 2018). arXiv : 1810.08760 (cf. p. 2).

[Cau02] Didier Caucal. « On Infinite Terms Having a Decidable Monadic Theory ». In : *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*. Sous la dir. de Krzysztof Diks et Wojciech Rytter. T. 2420. Lecture Notes in Computer Science. Springer, 2002, p. 165-176. doi : 10.1007/3-540-45687-2_13 (cf. p. 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. 6).

[CW03] Arnaud Carayol et Stefan Wöhrle. « The Caucal Hierarchy of Infinite Graphs in Terms of Logic and Higher-Order Pushdown Automata ». In : *FST TCS 2003 : Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*. Sous la dir. de Paritosh K. Pandya et Jaikumar Radhakrishnan. T. 2914. Lecture Notes in Computer Science. Springer, 2003, p. 112-123. doi : 10.1007/978-3-540-24597-1_10 (cf. p. 5).

[Dam82] Werner Damm. « The IO- and OI-Hierarchies ». In : *Theoretical Computer Science* 20 (1982), p. 95-207. doi : 10.1016/0304-3975(82)90009-3 (cf. p. 4).

[DJ03] Vincent Danos et Jean-Baptiste Joinet. « Linear logic and elementary time ». In : *Information and Computation*. International Workshop on Implicit Computational Complexity (ICC'99) 183.1 (mai 2003), p. 123-137. issn : 0890-5401. doi : 10.1016/S0890-5401(03)00010-5 (cf. p. 4).

[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. 6).

[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. 2, 3).

[EM03] Joost Engelfriet et Sebastian Maneth. « A comparison of pebble tree transducers with macro tree transducers ». In : *Acta Informatica* 39.9 (2003), p. 613-698. doi : 10.1007/s00236-003-0120-0 (cf. p. 2, 5).

[EV86] Joost Engelfriet et Heiko Vogler. « Pushdown Machines for the Macro Tree Transducer ». In : *Theoretical Computer Science* 42 (1986), p. 251-368. doi : 10.1016/0304-3975(86)90052-6 (cf. p. 2-5).

[EV88] Joost Engelfriet et Heiko Vogler. « High Level Tree Transducers and Iterated Pushdown Tree Transducers ». In : *Acta Informatica* 26.1/2 (1988), p. 131-192. doi : 10.1007/BF02915449 (cf. p. 3, 4).

[FLO83] Steven Fortune, Daniel Leivant et Michael O'Donnell. « The Expressiveness of Simple and Second-Order Type Structures ». In : *Journal of the ACM* 30.1 (jan. 1983), p. 151-185. issn : 0004-5411. doi : 10.1145/322358.322370 (cf. p. 3).

[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, 6).

[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).

[GMP24] Mateusz Gienieczko, Filip Murlak et Charles Paperman. *Supporting Descendants in SIMD-Accelerated JSONPath*. To appear in the proceedings of ASPLOS 2024. 2024. url : https://github.com/V0ldek/rsonpath/blob/main/pdf/supporting-descendants-in-simd-accelerated-jsonpath.pdf (cf. p. 2).

[Gre16] Charles Grellois. « Semantics of linear logic and higher-order model-checking ». en. Thèse de doct. Université Paris 7, avr. 2016. url : https://tel.archives-ouvertes.fr/tel-01311150/ (cf. p. 4).

[Hag+17] Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong et Olivier Serre. « Collapsible Pushdown Automata and Recursion Schemes ». In : *ACM Transactions on Computational Logic* 18.3 (août 2017), 25 :1-25 :42. issn : 1529-3785. doi : 10.1145/3091122 (cf. p. 4).

[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. 3, 5).

[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. 2).

[KNU02] Teodor Knapik, Damian Niwiński et Paweł Urzyczyn. « Higher-Order Pushdown Trees Are Easy ». In : *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8-12, 2002, Proceedings*. Sous la dir. de Mogens Nielsen et Uffe Engberg. T. 2303. Lecture Notes in Computer Science. Springer, 2002, p. 205-222. doi : 10.1007/3-540-45931-6_15 (cf. p. 4).

[Kob13] Naoki Kobayashi. « Pumping by Typing ». In : *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*. IEEE Computer Society, 2013, p. 398-407. doi : 10.1109/LICS.2013.46 (cf. p. 4).

[KTU10] Naoki Kobayashi, Naoshi Tabuchi et Hiroshi Unno. « Higher-order multi-parameter tree transducers and recursion schemes for program verification ». In : *Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2010, Madrid, Spain, January 17-23, 2010*. Sous la dir. de Manuel V. Hermenegildo et Jens Palsberg. ACM, 2010, p. 495-508. doi : 10.1145/1706299.1706355 (cf. p. 3).

[Lau09] Olivier Laurent. « On the categorical semantics of Elementary Linear Logic ». In : *Theory and Applications of Categories* 22.10 (juin 2009), p. 269-301. url : http://www.tac.mta.ca/tac/volumes/22/10/22-10.pdf (cf. p. 5).

[Lei93] Daniel Leivant. « Functions over free algebras definable in the simply typed lambda calculus ». In : *Theoretical Computer Science* 121.1 (déc. 1993), p. 309-321. issn : 0304-3975. doi : 10.1016/0304-3975(93)90092-8 (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, 6).

[MS13] Sebastian Maneth et Helmut Seidl. « Tree Transducers and Formal Methods (Dagstuhl Seminar 13192) ». In : *Dagstuhl Reports* 3.5 (2013), p. 1-18. issn : 2192-5283. doi : 10.4230/DagRep.3.5.1 (cf. p. 5).

[MSV03] Tova Milo, Dan Suciu et Victor Vianu. « Typechecking for XML transformers ». In : *Journal of Computer and System Sciences* 66.1 (2003). Journal version of a PODS 2000 paper, p. 66-97. doi : 10.1016/S0022-0000(02)00030-2 (cf. p. 2).

[Ngu19] 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. 4).

[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. 4).

[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. 4).

[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. 3).

[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).

[OR11] C.-H. Luke Ong et Steven J. Ramsay. « Verifying higher-order functional programs with pattern-matching algebraic data types ». In : *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*. Sous la dir. de Thomas Ball et Mooly Sagiv. ACM, 2011, p. 587-598. DOI : 10.1145/1926385.1926453 (cf. p. 3).

[Par20] Pawel Parys. « On the Expressive Power of Higher-Order Pushdown Systems ». In : *Logical Methods in Computer Science* 16.3 (2020). DOI : 10.23638/LMCS-16(3:11)2020 (cf. p. 4).

[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).

[Pin21] Laureline Pinault. « From automata to cyclic proofs : equivalence algorithms and descriptive complexity ». Thèse de doct. École normale supérieure de Lyon, juill. 2021. URL : https://tel.archives-ouvertes.fr/tel-03412556 (cf. p. 6).

[Plo22] Gordon Plotkin. *Recursion does not always help*. 2022. arXiv : 2206.08413 [cs.LO] (cf. p. 4).

[Pra20] Cécilia Pradic. « Some proof-theoretical approaches to Monadic Second-Order logic ». Thèse de doct. École normale supérieure de Lyon & Uniwersytet Warszawski (MIMUW), juin 2020. URL : https://tel.archives-ouvertes.fr/tel-02954006 (cf. p. 6).

[Ron18] Simona Ronchi Della Rocca. « Intersection Types and Denotational Semantics : An Extended Abstract ». In : *22nd International Conference on Types for Proofs and Programs (TYPES 2016)*. Sous la dir. de Silvia Ghilezan, Herman Geuvers et Jelena Ivetic. T. 97. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany : Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018, 2 :1-2 :7. ISBN : 978-3-95977-065-1. DOI : 10.4230/LIPIcs.TYPES.2016.2 (cf. p. 4).

[Rub17] Thomas Rubiano. « Implicit Computational Complexity and Compilers ». Thèse de doct. Université Sorbonne Paris Cité, déc. 2017. URL : https://tel.archives-ouvertes.fr/tel-02362912 (cf. p. 2).

[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).

[Soy23] Claire Soyez-Martin. « From semigroup theory to vectorization : recognizing regular languages ». en. Thèse de doct. Université de Lille, déc. 2023 (cf. p. 2).

[SW16] Sylvain Salvati et Igor Walukiewicz. « Simply typed fixpoint calculus and collapsible pushdown automata ». en. In : *Mathematical Structures in Computer Science* 26.7 (oct. 2016), p. 1304-1350. ISSN : 0960-1295, 1469-8072. DOI : 10.1017/S0960129514000590 (cf. p. 4).

[Zai91] Marek Zaionc. « λ-Definability on free algebras ». In : *Annals of Pure and Applied Logic* 51.3 (mars 1991), p. 279-300. ISSN : 0168-0072. DOI : 10.1016/0168-0072(91)90019-I (cf. p. 3).