Projet de recherche : Étude de classes canoniques de transductions provenant de λ-calculs

Lê Thành Dũng (Tito) Nguyễn – postdoc Milyon 2022

Contexte et résumé du projet

Comparer le pouvoir expressif de différents modèles de calcul – par des équivalences, inclusions, séparations… – est l'une des préoccupations initiales de l'informatique théorique, restée centrale dans la recherche contemporaine : en témoigne par exemple l'essor de la théorie de la complexité, ou encore de la *théorie des automates*. Mais à côté du « pouvoir calculatoire », un autre courant majeur de l'informatique fondamentale concerne la « structure des programmes ». (Les termes « pouvoir » et « structure » pour désigner cette dichotomie sont empruntés à [AS21].) Il s'agit par exemple de comprendre comment définir précisément le sens des programmes, les analyser, et/ou les construire… Le tout de façon modulaire, d'où le succès qu'y a rencontré la théorie des catégories, langage mathématique de la compositionnalité, alors qu'elle semble inopérante jusqu'à présent en algorithmique ou en complexité. La théorie contemporaine des langages de programmation – en particulier *l'étude du λ-calcul et de la théorie des types* – s'inscrit dans ce courant « structure ».

Mon projet de recherche se concentre en grande partie, de façon inhabituelle, sur les aspects « pouvoir calculatoire » du λ-calcul, en creusant des liens avec les automates. Ceci implique de se situer à l'interface entre deux communautés qui sont traditionnellement très distinctes, même si elles se parlent de plus en plus. Il s'agit notamment, dans le prolongement de ma thèse [Ngu21] :

- d'employer les λ-calculs typés comme source de nouveaux objets intéressants pour la théorie des automates – en évitant de simplement introduire des définitions arbitraires, c'est-à-dire avec une attention particulière portée au caractère robuste et canonique de ces objets ;
- de répondre ainsi à des questions qui, indépendamment de la simple envie de relier automates et λ-calculs entre eux, se poseraient dans chacun de ces deux domaines.

Quelques détails

Une exception importante depuis 30 ans à la séparation structure/pouvoir est le sous-domaine de la *complexité implicite*. Rappelons que les langages de programmation impliqués dans la célèbre correspondance preuves-programmes (ou « isomorphisme de Curry–Howard » garantissent souvent que les programmes *terminent* car cela implique, côté preuves, la cohérence logique. Les λ-calculs typés sont des exemples typiques de tels langages de programmation. Une garantie de terminaison fournie par un système de types peut même venir avec des bornes quantitatives de complexité en temps. Ceci mène à l'idée de caractériser des classes de complexité en employant des langages « de haut niveau », sans mentionner explicitement des bornes en temps ou en espace.

Mes travaux menés en thèse avec Pierre Pradic sur les « automates implicites » constituent une variante sur ce thème où l'on cherche à montrer que le pouvoir calculatoire d'un λ-calcul typé correspond non pas à une classe de complexité, mais à un modèle d'automates. Néanmoins, nos recherches se distinguent de la complexité implicite par quelques aspects :

- Certaines de nos équivalences en pouvoir expressif sont obtenues comme conséquences de liens structurels plus profonds, exprimés dans le langage des catégories [Ngu21, §1.2.3]. Moralement, c'est possible car contrairement à la complexité algorithmique, la théorie des automates a depuis longtemps adopté une vision compositionnelle du calcul, exprimée algébriquement (e.g. par des semigroupes finis), cf. [Ngu21, §1.1.2]. Il est d'ailleurs habituel d'appliquer cette « structure » à des questions de « pouvoir » calculatoire des automates !

- Ceci fonctionne aussi car nous travaillons dans des λ-calculs dont nous connaissons la bonne notion de sémantique catégorique, puisqu'ils sont définis en combinant des fonctionnalités standard (fonctions linéaires/affines, connecteurs additifs, non-commutativité). Autrement dit, nous nous efforçons d'obtenir des résultats « naturels », sans choix de conception ad-hoc pour tenter de caractériser une classe de fonctions voulue – ce qui est inhabituel en complexité implicite.

- Grâce au point précédent, quand nous sommes tombés sur une classe qui n'était pas encore connue en étudiant l'expressivité d'un λ-calcul linéaire, il était raisonnable d'espérer qu'elle soit intéressante pour d'autres raisons. Ce fut en effet le cas [Ngu21, §1.2.4] : il y avait suffisamment à dire sur ces fonctions, baptisées « polyrégulières sans comparaison », du point de vue de la pure théorie des automates pour faire l'objet d'un article sans λ-termes à ICALP [NNP21].

Ainsi, alors que nous ne cherchions au départ qu'à caractériser des classes de fonctions préexistantes, mon projet vise désormais à **employer les λ-calculs typés comme source de nouveaux objets intéressants pour la théorie des automates** – en particulier des *transductions* (fonctions calculées par des transducteurs i.e. automates à sortie) telles que les fonctions polyrégulières sans comparaison. La découverte de ces dernières est un signe prometteur.

Certes, d'autres motivations côté automates auraient également pu mener à les introduire. S'agissant d'une nouvelle classe de fonctions, avoir plusieurs caractérisations de natures très différentes est toutefois rassurant (et cela est d'autant plus convaincant qu'on s'efforce d'avoir des définitions naturelles, d'où l'importance du second item ci-dessus). En effet, c'est ainsi que l'on repère les notions à peu près *robustes* et *canoniques* parmi l'infinie diversité des formalismes qu'on pourrait en principe étudier – car nul ne saurait définir rigoureusement ce qu'est un « bon » modèle de calcul… Dans « objets intéressants », j'inclus donc l'espoir d'un semblant de canonicité.

Un autre intérêt visé (en fait relié au précédent) réside en la capacité de ces objets à **répondre à des questions qui, indépendamment de la simple envie de relier automates et λ-calculs entre eux, se poseraient dans chacun de ces deux domaines**. C'est ce que je vais tenter d'illustrer par trois pistes concrètes tirées de la section « travaux en cours » de mon manuscrit de thèse [Ngu21, §1.4], chacune contenant une portion « à faible risque » et une autre plus spéculative.

- La première concerne les fonctions définissables en λ-calcul simplement typé. Il peut paraître étonnant qu'une vieille question (datant au moins des années 1980) à ce propos soit encore ouverte ; c'est sans doute dû en partie à la division sociologique entre les communautés qui s'occupent de « structure » et celles qui s'intéressent au « pouvoir ».

- De même, la seconde part de l'étude de l'expressivité d'un λ-calcul affine, motivation initiale qui ne parle pas d'automates (mais ceux-ci feront a priori partie des réponses).

- Enfin, dans la dernière, je propose de faire un détour par le λ-calcul pour trouver les « bonnes » généralisations de certaines classes de langages et fonctions. Une fois ces définitions posées, elles pourront guider la conception de modèles d'automates adéquats.

À chacune de ces pistes, je dédie l'une des sections qui suit, dans l'ordre.

Transductions en λ-calcul simplement typé [Ngu21, §1.4.1]

Quelles sont les fonctions $\mathbb{N}^k \to \mathbb{N}$ définissables par des λ-termes (programmes) simplement typés, en utilisant la représentation habituelle (codage de Church) des entiers naturels ? Pour une interprétation de cette question, on connaît la réponse depuis longtemps [Sch75], ce sont les « polynômes étendus ». Par contre, aucune caractérisation satisfaisante n'est connue pour une autre interprétation, où l'on autorise des substitutions dans le type d'entrée – c'est une forme habituelle de polymorphisme, présente notamment dans un résultat fondamental [HK96, Th. 4.3] d'où partent mes travaux de thèse. L'application de ce théorème aux entiers (vus comme mots unaires) explique a posteriori des résultats d'inexpressivité bien connus des années 1980 – on peut coder des fonctions à croissance hyperexponentielle, mais pas la soustraction ou l'égalité ! – par des propriétés des langages rationnels, cf. [Ngu21, §1.3.5].

Je propose de chercher un *modèle d'automates* qui réponde à cette question – en fait, à une généralisation où l'on considère non pas des fonctions sur les entiers, mais sur les arbres. En effet, d'après Hillebrand et Kanellakis, les fonctions λ-définissables doivent préserver les langages rationnels d'arbres par image réciproque, ce qui semble indiquer qu'on a affaire à des automates.

En fait, pour un sous-système du λ-calcul simplement typé appelé λ-calcul sûr (*safe*) [BO09], une réponse est à portée de main (il ne semble pas y avoir de difficulté technique majeure) : il s'agit d'une classe de transductions d'arbres étudiée intensivement par Engelfriet et al., que j'appellerai ici $\mathcal{E}$. Parmi les définitions équivalentes de $\mathcal{E}$ connues, on en trouve deux [EV88 ; EV86] qui sont très proches des automates à piles d'ordre supérieur et des schémas de récursion sûrs – et une leçon des travaux des 20 dernières années en *model-checking d'ordre supérieur* (cf. [Ngu21, §1.3.4]), autre domaine reliant automates et λ-calcul, est que ces deux notions sont reliées de près au λ-calcul sûr.

Le model-checking d'ordre supérieur nous dit aussi qu'enlever la condition de sûreté du λ-calcul et des schémas de récursion correspond à rajouter une opération de « collapse » aux automates à piles d'ordre supérieur. On peut donc espérer en tirer une notion de « transducteur d'arbres à collapse » qui correspondrait au pouvoir expressif du λ-calcul simplement typé. Cependant, ces transducteurs n'existent pas encore dans la littérature, donc ceci devrait nécessiter plus de travail.

Enfin, si cela s'avère fonctionner, il serait intéressant de mieux comprendre cette classe, une fois sa canonicité établie. Ainsi, dans le cas « sûr », on sait que toute fonction de $\mathcal{E}$ peut s'écrire comme composition de transducteurs d'arbres beaucoup plus « simples » (transducteurs macro, à jetons, ou cheminants, cf. [EV86 ; EM03]). De tels théorèmes de factorisation peuvent-ils exister pour les fonctions d'arbres définissables en λ-calcul simplement typé ? On pourrait aussi chercher à caractériser, parmi ces fonctions, celles qui sont à croissance linéaire (fait pour $\mathcal{E}$ dans [EIM21]) ou polynomiale (cf. [Ngu21, Conjecture 1.4.14]).

Langages et fonctions d'arbres en λ-calcul affine sans additifs [Ngu21, §1.4.2–1.4.3]

Les caractérisations implicites de transductions dans ma thèse s'appuient sur un λ-calcul linéaire *avec connecteurs additifs*, ces derniers présentant des liens étroits avec les fonctionnalités de certains transducteurs d'arbres [Ngu21, §1.2.2]. Il est naturel de se demander ce qu'il arrive alors si on enlève ces connecteurs. En l'absence d'additifs, comme l'indique [Ngu21, Th. 7.0.2], il est alors préférable de considérer un typage *affine* plutôt que linéaire (la différence étant qu'une fonction linéaire (resp. affine) utilise son argument exactement (resp. au plus) une fois).

Dans le cas des fonctions des mots vers les mots, nous savons déjà démontrer qu'enlever les additifs ne diminue pas l'expressivité. La preuve n'est pas encore écrite, mais le plus dur est déjà fait dans [NP20, Th. 5.4] qui code les fonctions séquentielles (le passage aux fonctions régulières, puis polyrégulières sans comparaison, utilise [Ngu21, Th. 1.4.7] dû à Bojańczyk et al.). De plus, la caractérisation des langages sans étoile (aussi appelés apériodiques) que Pradic et moi avons obtenu dans [NP20] en λ-calcul non commutatif dépend de façon cruciale de l'absence d'additifs (cf. [NP20, §5.2]), ce qui ouvre la porte à des caractérisations de transductions apériodiques.

Le cas des arbres est moins routinier. Dans le cas des langages (ou, cela revient au même, des fonctions renvoyant un booléen), les connecteurs additifs semblent déjà avoir un effet sur le pouvoir calculatoire. Avec les additifs, on peut facilement écrire des λ-termes affines (voire même linéaires) calculant n'importe quel langage rationnel d'arbre ; sans eux, la classe de langages obtenue devrait être incluse dans celle définie par les *automates cheminants* (*tree-walking*), or ces derniers ne savent pas reconnaître certains langages rationnels [BC08]. La stratégie de preuve que je prévois fait intervenir une approche catégorique semblable à celle de ma thèse, passant par la sémantique des jeux affine. Cette dernière est fondamentalement liée à la « géométrie de l'interaction » qui à son tour correspond aux automates cheminants du point de vue de la théorie catégorique des automates. Un cas particulier de cette correspondance (sur les mots) a déjà servi de source d'inspiration pour un modèle d'automates, proposé par Hines et étudié par Pradic et moi [NP21].

Reste qu'à ma connaissance, l'inclusion entre λ-calcul affine et automates cheminants n'a pas de raison a priori d'être une égalité. D'où une question plus exploratoire : quelle classe de langages d'arbres définit-on en fait avec le λ-calcul affine ? Et quelles transductions d'arbres ? Ces dernières ont automatiquement quelques bonnes propriétés en vertu de leur définition par λ-calcul : par exemple, elles forment une sous-classe stable par composition des fonctions régulières d'arbres. Le triptyque λ-calcul / schémas de récursion / modèle d'automates a été étudié récemment dans le cas purement affine sans additifs [CM19], ce qui pourrait constituer une source d'inspiration.

Alphabets infinis : des mots aux arbres [Ngu21, §1.4.4]

Il s'agit ici de considérer des mots ou des arbres sur des alphabets faisant intervenir un ensemble infini d'*atomes* $\mathbb{A}$; dans le cas le plus simple, l'alphabet pourra être de la forme $\Sigma \times \mathbb{A}$ où Σ est un ensemble fini d'étiquettes. Moralement, les atomes ne sont pas censés avoir de structure interne, la seule chose qu'on puisse faire étant de les stocker, les copier ou les soumettre à un test d'égalité (formellement, cela s'exprime par des propriétés d'invariance par l'action de bijections $\mathbb{A} \to \mathbb{A}$).

L'étude de ce thème en théorie des automates est ancienne, mais les questions de robustesse et canonicité s'y posent de façon aiguë : pour citer Bojańczyk et Stefański [BS20], « The literature for infinite alphabets is full of depressing diagrams [...] which describe countless models that satisfy only trivial relationships ». La contribution majeure de [BS20] est de remédier à cet état de fait, en dégageant des généralisations des langages rationnels et fonctions régulières habituels aux mots sur des alphabets infinis qui admettent de nombreuses définitions équivalentes – parmi lesquelles des modèles machines sujets à des restrictions dites de « single use », c'est-à-dire : de linéarité.

L'idée de transposer les résultats d'automates implicites de ma thèse, à base de λ-calcul linéaire, au cadre des alphabets infinis apparaît donc évidente. Nous nous sommes convaincus, avec Clovis Eberhart et Pierre Pradic, que ça devrait marcher : la transposition naïve devrait bien nous mener à caractériser les classes étudiées dans [BS20]. Cependant, les techniques catégoriques de ma thèse ne fonctionnent plus, et nous avons dû nous résoudre à attaquer le problème par des méthodes syntaxiques moins éclairantes structurellement. Même si le théorème concret est, a priori, celui auquel l'on s'attendait, il reste donc malgré tout une facette incomprise dans cette histoire.

Une question plus ouverte consiste à étendre tout cela aux arbres. Cette généralisation n'est pas connue côté automates ! Une obstruction est que la notion de « single use » dans [BS20] est « sans additifs », proche des « streaming string transducers » *sans copie* – or si cela convient sur les mots, ce n'est plus le cas pour les arbres, et on l'observe déjà sur les alphabets finis [Ngu21, §1.2.2]. Et adapter la « single use restriction avec additifs » des transducteurs d'arbres [EM99 ; AD17] est loin d'être évident, à cause de l'influence que les atomes stockés en mémoire peuvent avoir sur le flot de contrôle – c'est le même phénomène qui « casse » nos méthodes catégoriques.

Par contre, avec le λ-calcul (linéaire avec additifs), rien de plus simple : il suffit de remplacer le codage des mots par celui des arbres ! Ainsi, le λ-calcul devrait fournir une généralisation naturelle des langages et fonctions de [BS20] aux arbres sur alphabets infinis, qu'on peut espérer être la « bonne ». Une mise à l'épreuve de cette idée serait de vérifier qu'on retombe bien sur les langages

d'arbres définis par le formalisme logique de [CLP15], qui caractérise les mêmes langages de mots que [BS20] et permet aussi un passage simple aux arbres. Pour les fonctions, passer par le λ-calcul est le seul moyen que je connaisse de tenter de définir de façon convaincante la notion de « fonction régulière d'arbres sur alphabets infinis » – une question définitionnelle qui présente un intérêt intrinsèque pour la théorie des automates, indépendamment du λ-calcul. Il faudra ensuite voir si l'on peut concevoir, pour ces fonctions, des modèles de transducteurs satisfaisants, ou encore les engendrer à partir d'un petit nombre de fonctions primitives à la manière de [BD20]…

Références

[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. 4).

[AS21] Samson Abramsky et Nihil Shah. « Relating structure and power : Comonadic semantics for computational resources ». In : *Journal of Logic and Computation* 31.6 (2021). Long version of a CSL'18 paper, p. 1390-1428. doi : 10.1093/logcom/exab048 (cf. p. 1).

[BC08] Mikołaj Bojańczyk et Thomas Colcombet. « Tree-walking automata do not recognize all regular languages ». In : *SIAM Journal on Computing* 38.2 (2008), p. 658-701. doi : 10.1137/050645427 (cf. p. 4).

[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. 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. 3).

[BS20] Mikołaj Bojańczyk et Rafał Stefański. « Single-Use Automata and Transducers for Infinite Alphabets ». 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, 113 :1-113 :14. doi : 10.4230/LIPIcs.ICALP.2020.113 (cf. p. 4, 5).

[CLP15] Thomas Colcombet, Clemens Ley et Gabriele Puppis. « Logics with rigidly guarded data tests ». In : *Logical Methods in Computer Science* 11.3 (2015). doi : 10.2168/LMCS-11(3:10)2015 (cf. p. 5).

[CM19] Pierre Clairambault et Andrzej S. Murawski. « On the Expressivity of Linear Recursion Schemes ». In : *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*. Sous la dir. de Peter Rossmanith, Pinar Heggernes et Joost-Pieter Katoen. T. 138. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 50 :1-50 :14. isbn : 978-3-95977-117-7. doi : 10.4230/LIPIcs.MFCS.2019.50 (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. 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. 3).

[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. 4).

[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. 3).

[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).

[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).

[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://nguyentito.eu/thesis.pdf (cf. p. 1-4).

[NNP21] Lê Thành Dũng Nguyễn, Camille Noûs et Pierre 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. 2).

[NP20] Lê Thành Dũng Nguyễn et Pierre 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).

[NP21] Lê Thành Dũng Nguyễn et Pierre Pradic. « The planar geometry of first-order transductions ». In preparation. Slides available at https://nguyentito.eu/2021-01-links.pdf. 2021.

[Sch75] Helmut Schwichtenberg. « Definierbare Funktionen im λ-Kalkül mit Typen ». de. In : *Archiv für mathematische Logik und Grundlagenforschung* 17.3 (sept. 1975), p. 113-114. ISSN : 1432-0665. DOI : 10.1007/BF02276799 (cf. p. 3).