Logique linéaire et classes de complexité sous-polynomiales

Rapport de soutenance à mi-parcours

Clément Aubert [*] [†]

29 juin 2012

Directeur :	Stefano Guerini, LIPN
Co-encadrant :	Virgile Mogbil, LIPN
Invités extérieurs au laboratoire :	Patrick Baillot, CNRS - LIP
	Paul-André Melliès, CNRS - PPS
Invitée interne au laboratoire :	Laure Petrucci, LIPN
Directeur de laboratoire :	Christophe Fouqueré, LIPN

1 Introduction

Dans le domaine de la théorie de la démonstration, la *logique linéaire* (**LL**) – du fait de sa sensibilité aux ressources employées lors de l'établissement d'une preuve – est un formalisme privilégié pour aborder les questions de complexité. La correspondance de Curry-Howard, qui s'étend naturellement à des considérations quantitatives, nous enseigne que sa dynamique d'évaluation – l'élimination des coupures – est isomorphe à l'exécution d'un programme rédigé en langage abstrait avec spécification. La théorie de la complexité s'attache quant à elle à déterminer si un problème admet une solution, et le cas échéant borne les ressources en temps et en espace nécessaires à l'établissement d'un résultat. Une *machine de Turing déterministe* (resp. *non-déterministe*) employant un espace logarithmique en la taille de son ruban d'entrée est dite *logarithmique* et définit la classe de complexité

[*] Université Paris 13, Sorbonne Paris Cité, LIPN, F-93430, Villetaneuse, France.
[†] CNRS, UMR 7030, F-93430, Villetaneuse, France.

L (resp. NL), alors qu'une machine utilisant un temps polynomial – indépendamment du degré maximal du polynôme – en la taille de son ruban d'entrée est dite *poly-time* et définit la classe P. Ces mesures – théoriques – sont parfois très sensibles au formalisme employé, à la modélisation abstraite de la machine et à la représentation des données. Les classes de complexité les plus robustes permettent de se dégager des modèles pour se focaliser sur les problèmes ; on donne ci-dessous la hiérarchie des classes qui seront utiles dans le cadre de cet exposé [1] :

$$NC^0 \subseteq AC^0 \subsetneq NC^1 \subseteq L \subseteq NL \subseteq AC^1 \subseteq \ldots \subseteq AC = NC \subseteq P$$

Les premiers travaux visant à établir une correspondance entre classes de complexité et **LL** ont eu lieu dans le cadre de la complexité implicite, dont les trois approches principales sont les restrictions du schéma de récursion, les systèmes de réécriture avec quasi-interprétation et la **LL**. En ce qui concerne la théorie de la démonstration, il s'agit de définir des sous-systèmes de la **LL** munis de mécaniques d'évaluation bridées dans lesquelles tous les programmes exprimables ont une complexité bornée. Les premiers travaux en ce sens remontent à l'invention des logiques linéaires dites *légères* (**LLL**), qui reposent sur le contrôle de la duplication et de l'effacement, responsables de la complexité de l'élimination des coupures et gérées au moyens de modalités (! et son dual ?). Les premières découvertes remontent à [Gir95] qui introduisait une nouvelle modalité (§) et contraignait l'évaluation à avoir lieu par profondeur. D'autres travaux ([DJ03], [Laf04] et [BM10] notamment) suivirent et permirent notamment de se dispenser de §, d'introduire l'évaluation par niveaux ou de capturer de plus en plus extensionnellement des classes de complexité égales à P – classe souvent considérée comme celle des problèmes solubles en temps raisonnable – ou supérieures.

D'autres perspectives ont été ouvertes par les travaux de Martin Hofmann [Hof03] – qui peuvent facilement s'adapter à la **LL** –, de Ugo Dal Lago [Lag05] – qui emploie des outils sémantiques – ou encore de Patrick Baillot et Marco Pedicini [BP01] – qui utilisent la Géométrie de l'Interaction (**GdI**).

Seulement, hormis **SBAL** de Ulrich Schöpp [Sch07] qui borne les ! et les $\forall$ avec un polynôme de ressource et caractérise ainsi L, toutes ces restrictions ne permettent d'obtenir que des systèmes de complexité égale ou supérieure à P. Plus grave, le non-déterminisme n'est que simulé par la juxtaposition de calculs déterministes et n'a pas de véritable caractérisation logique, et le parallélisme est tout simplement absent du fait de la stratification des **LLL**.

1. Les définitions de NC^i et AC^i seront rappelées au début de la section suivante, notons juste pour le moment que par définition $\cup_{i\in\mathbb{N}} AC^i = AC$ et $\cup_{i\in\mathbb{N}} NC^i = NC$.

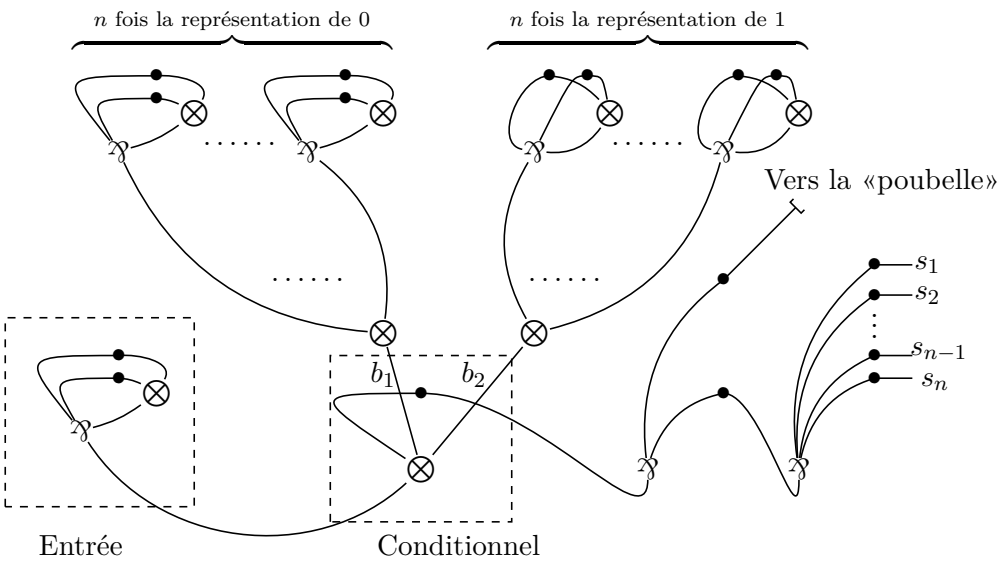
Notre travail tente de combler ces lacunes, afin d'enrichir le paysage de la complexité et ses rapports à la **LL** par la représentation de nouvelles classes (sous-polynomiales, parallèles et non-déterministes). En diversifiant les approches, nous espérons être capables de représenter une multitude de classes dans un formalisme unifié inspiré de la **Gdi**. Cette approche a connu une modification si forte entre [Gir89] et [Gir11a] qu'il est usuel de dire qu'il en existe deux versions, toutes deux profitant des bonnes propriété des algèbres d'opérateurs. La plus ancienne interprétait la terminaison du calcul comme la nilpotence de matrices issues de C^*-algèbres, et permit la définition d'une formule mathématique représentant la normalisation de réseaux de preuves, la *formule d'exécution*. La dernière version élargit ce cadre en ayant recours aux algèbres de von Neumann, qui permettent de remplacer la nilpotence par l'annulation du déterminant. Cependant ce cadre – plus riche – ne se raccorde que de façon très imparfaite à la notion de *chemins*, qui avait permis une simplification de la **GdI** dans son ancienne version et de la relier aux fondements de l'informatique *via* le λ-calcul et les graphes de partage.

Le travail mené jusqu'ici a pour point de départ deux travaux, [Ter04] et [Gir11b] : ils permettent respectivement de représenter à l'aide de la **LL** des classes parallèles (NC et AC, section 2) et non-déterministes (NL, section 3).

2 Réseaux de preuves pour la logique linéaire et parallélisme [2]

Les classes NC^i et AC^i ont étés initialement définies pour exprimer la complexité des circuits booléens de profondeur polylogarithmiques (d'ordre $(\log(n))^i$) et de taille polynomiale en la taille de l'entrée, avec des portes exécutant des opérations booléennes binaires pour NC^i et d'arité quelconque pour AC^i. Ces classes sont généralement présentées comme les classes de problèmes qui peuvent être traités plus efficacement en parallèle que séquentiellement [Coo85]. De plus, NC^0, AC^0 et NC^1 – en tant que classes sous-logarithmiques – sont d'un intérêt majeur car elles offrent un cadre adapté aux réductions de problèmes vers L. Les deux façons les plus courantes de les modéliser sont les circuits booléens (voir [Vol99] pour un bon état des lieux) et les machines de Turing alternantes (**MTA**) [CS76]. Malgré le caractère éminemment parallèle des réseaux de preuves, il n'existait pas

2. Travaux soutenu par l'ANR Blanc Complice, `http://www-lipn.univ-paris13.fr/complice/`



Le calcul débute par la réduction de la coupure entre l'entrée fournie et l'entrée du conditionnel, puis se propage au réseau en parallèle. La branche b_1 ou b_2, en fonction de la planarité de l'entrée, est évacuée vers la poubelle, et l'autre «paquet de valeurs» est ouvert par la coupure avec le $\parr$ et propagé aux fils $s_1, \ldots, s_n$.

FIGURE 1 – Duplication d'une valeur n fois

avant 2004 de caractérisation en **LL** de ces classes de complexité. Cela vient du fait que les logiques légères tirent leurs bornes d'une évaluation «stratifiée» (par niveaux ou par «boîtes») des réseaux, qui interdit un partage ou une parallélisation des calculs intermédiaires.

[Aub11] présente une simplification des travaux de [Ter04], qui introduisait une façon innovante de borner la complexité d'un système logique avec les *réseaux de preuve booléens* : duplications et effacements ne sont plus gérées au moyen des modalités mais simulées «en dur», dans la structure même des réseaux de preuve. On illustre les principaux mécanismes de ce formalisme en représentant à la figure 1 la duplication n fois d'une valeur passée en entrée. Le cœur calculatoire de ce formalisme est le conditionnel, qui simule le `if-then-else` et permet de simuler la conjonction et la disjonction.

La taille du réseau – son nombre de connecteurs et d'axiomes – et sa

profondeur – la taille de la plus grande formule coupée – sont deux mesures de ces *circuits de preuves*. En se restreignant aux profondeurs polylogarithmiques et aux tailles polynomiales, on établit naturellement une correspondance avec les circuits booléens. L'évaluation est réalisée au moyen d'une parallélisation efficace de l'élimination des coupures, qui introduit la *tightening reduction* pour gérer en parallèle les coupures-axiomes. Du fait de la gestion «en dur» de l'effacement et de la duplication, notre réseau est adapté à une taille d'entrée donnée, à la manière des circuits booléens.

Cette particularité nous force à considérer des questions d'uniformité [Ruz81], ignorées par Kazushige Terui et résolues par Virgile Mogbil dans [MR07]. Dans [Aub11] j'ai repris ce cadre uniforme et ai fortement allégé les réseaux booléens en inventant les *pièces*, qui simulent les briques élémentaires du calcul, facilitent la construction de réseaux de preuves booléens et réduisent leur taille. Du fait de cette simplification, la traduction des circuits booléens en réseaux – qui consommait auparavant des ressources d'ordre logarithmique – peut désormais être accomplie avec des circuits de profondeur constante. Outre DICE 2011 [3], j'ai également eu l'occasion de présenter ces travaux aux 16ème Rencontres LAC du GdR-IM [4] et en séminaire à l'Institut mathématiques de Luminy [5].

Ces travaux pourraient faire l'objet d'une publication en journal, afin de clarifier les relations de ce modèle aux **MTA** [6] : la simulation des **MTA** par circuits de preuves se fait en adaptant une preuve de [Vol99], mais la simulation réciproque pose nettement plus de problèmes. La puissance de l'alternance des **MTA** – qui permets notamment de faire des suppositions sur les entrées plutôt que d'aller les lire – ne suffit pas à compenser les limitations logarithmiques en espace, et une procédure de simulation efficace des réseaux de preuves est délicate. Un autre devenir possible est d'adapter ces circuits de preuves à l'uniformité rationnelle (*RAT*, [BM12]) afin d'obtenir une caractérisation plus fine : l'uniformité *DLOGTIME* employée jusqu'ici déploie des capacités de calcul supérieures à ce que peuvent fournir les circuits de profondeur constante ainsi construits.

Ces devenirs permettraient d'ancrer plus solidement les circuits de preuves comme modèle pertinent du parallélisme et d'enrichir la compréhension de l'évaluation des réseaux de preuves comme parcours de ceux-ci.

3. *Workshop* satellite de ETAPS, `http://dice11.loria.fr/DICE_2011/`

4. `http://www.pps.univ-paris-diderot.fr/~saurin/GT-LAC/`

5. `http://iml.univ-mrs.fr/ldp/Seminaire/`

6. Problème abordé lors de mon exposé à «Logic and Interaction 2012», pendant la semaine consacrée à la complexité, `http://li2012.univ-mrs.fr/`.

3 Les entiers Logspace : caractérisation logique du non-déterminisme

Jean-Yves Girard a récemment proposé dans [Gir11b] une lecture tout à fait innovante des questions de complexité en rapport avec la **LL** et la **GdI**, qui permets d'expliquer l'approche géométrique de l'informatique à l'aide d'outils algébriques. Si les réseaux de preuve permettent de s'affranchir de la bureaucratie de la séquentialité et d'effectuer plusieurs opérations en parallèle, un pas supplémentaire dans l'abstraction peut être réalisé. Partant du constat que tous les réseaux de preuve de même conclusion sont isomorphes à leurs axiomes près, un réseau est intégralement représenté par sa conclusion et la façon dont sont reliés ses axiomes. À partir de là, il est possible de représenter les réseaux de preuves avec de simples matrices : on numérote les occurrences des formules figurant dans les axiomes, et une matrice L à coefficients non nul en $L_{i,j}$ s'il y a un lien entre les occurrences numérotées i et j représente ce réseau. On ouvre ainsi la porte des algèbres d'opérateurs comme moyen de représenter les preuves et la mécanique de l'élimination des coupures, l'exécution pouvant être exprimée par des moyens purement algébriques (la formule d'exécution).

Il est ainsi possible de représenter dans un cadre commun (une algèbre de von Neumann de type II_∞, le facteur hyperfini) toute preuve représentant le codage binaire d'un entier. On représente ensuite l'algorithme dans le même formalisme, et on contraint légèrement les sous-algèbres afin de s'assurer que l'entrée et le programme peuvent interagir. Lorsque l'entrée est fixée, on se ramène à un cadre finitaire et on peut lancer l'exécution du calcul sur des matrices finies, les lois mathématiques effectuant alors le calcul d'une façon qui peut être interprétée comme non-déterministe : chaque base possible de la matrice à un moment donné de l'exécution représente une branche du calcul, qui se termine lorsque la matrice s'annule. L'annulation du déterminant — qui corresponds à l'aboutissement de la normalisation de la preuve en **GdI** — est ainsi identifiée avec la nilpotence de la matrice obtenue en multipliant la représentation de l'entier avec la représentation de l'algorithme. Son degré de nilpotence peut être interprété comme le temps nécessaire à l'obtention du résultat et peut sans doute être fixé assez finement.

On pose ainsi une contrainte sur la représentation des données en entiers qui nous assure que tout calcul ayant cette représentation des entiers comme entrée est dans NL – la classe des problèmes solubles par machines de Turing non-déterministes avec espace logarithmique. Ce cadre mathématique de définition d'une classe de complexité ouvre des perspectives prometteuses : par

exemple l'équivalence entre représentation des entrées n'est plus obtenues à l'aide d'algorithmes de codage car notre cadre de calcul est insensible aux représentations [7], qui sont toutes isomorphiques [8]. De même, la question de l'uniformité ne se pose plus, puisque notre programme est donné par un opérateur qui s'adapte par nature à l'entier fourni en entrée. Des restrictions de ce cadre afin d'obtenir l'espace logarithmique déterministe sont actuellement en cours de rédaction et pourraient s'avérer fructueuses : en établissant des relations logiques et mathématiques entre classes de complexité, leur séparation pourrait s'avérer plus aisée que dans le cadre conceptuel des machines de Turing. Une autre piste serait de contraindre la sous-algèbre afin de limiter l'expressivité des algorithmes. La **GdI** fournirait alors un cadre uniforme pour exprimer diverses classes de complexité.

Ce travail [9] en cours de rédaction avec Thomas Seiller [10] devrait faire l'objet d'une publication l'année prochaine.

Là où Girard affirme que son système simule des pointeurs et donc représente NL, nous proposons un algorithme pour **st-CONN-Comp** – un problème co-NL-complet– qui entérine de façon algorithmique ce résultat. Il pourrait cependant s'avérer fructueux d'approfondir la présentation de ce formalisme comme simple manipulation de pointeurs, afin de mieux comprendre les rapports de **PURPLE** [SH08] avec L. En effet ce système de pointeurs purs peut résoudre des problèmes L-complets tout en étant strictement inclus dans L, et notre cadre d'approche pourrait permettre de comprendre comment cette relation s'étend au non-déterminisme. Enfin, ce formalisme se situe à la frontière des deux pendants de la **GdI** : selon que nous options pour une algèbre de von Neumann ou un espace de Hilbert, nous retenons comme mécanisme de l'évaluation un critère de nilpotence ou d'annulation du déterminant. Nous pouvons ainsi librement alterner entre une conception mathématiques et une conception combinatoire du calcul et des classes de complexité. Nous pourrions ainsi rendre plus lisible un papier difficile et technique, prouver d'une nouvelle façon ce résultat et l'étendre.

7. Contrairement aux machines de Turing, où une représentation unaire des données fait voler en éclat les classes de complexité.

8. Il existe une infinité dénombrable de plongements — tous isomorphes — des matrices codant les réseaux de preuve vers une algèbre de von Neumann.

9. Travail qui a déjà bénéficié du programme «visite de doctorant» du GdR IM (`http://www.gdr-im.fr/?q=node/24`) et est partiellement soutenu par l'A.N.R. Blanc LOGOI (`http://logoi.fr/`)

10. Laboratoire de Mathématiques (LAMA) CNRS : UMR5127 – Université de Savoie, `http://iml.univ-mrs.fr/~seiller/`

4 Vers les graphes de partage

Il s'agira dans nos futurs travaux de continuer à privilégier comme objet d'étude les classes sous-polynomiales, ce pour quoi le λ-calcul offre un cadre idéal car il permet de conserver une approche logique et graphique de ces questions. Outre le fait que le λ-calcul est le liant «naturel» entre **LL**, exécution d'un programme et complexité, nous pouvons partir du constat simple que les problèmes complets pour L et NL sont des problèmes relatifs aux graphes. Les algorithmes «naturels» de ces classes de complexité étant des parcours de graphes (dirigés, non-dirigés, connexes ou non, etc.), il paraît sensé de tenter de représenter ces algorithmes au moyen de graphes de partage, qui sont des représentations graphiques du λ-calcul optimisant la mise en commun des ressources [AG98] et laissant entrevoir une possible caractérisation de classes parallèles. De plus, la version la plus traditionnelle de la **GdI** se ramène à un parcours de chemins dans des graphes, et permettrait de repasser des opérateurs à un système plus combinatoire. On pourrait alors tenter de reformuler la **GdI** 5 [Gir11a] de façon combinatoire, en suivant les travaux de Thomas Seiller [Sei12], et en essayant d'adopter comme formalisme les graphes de partage.

Nous pourrions ainsi – en nous rapprochant des langages de programmations – essayer de contourner l'écueil de nombreux travaux de complexité, qui utilisent des représentations des entrées très travaillées et ne peuvent implémenter simplement les constructeurs et destructeurs élémentaires. Une autre piste serait de cheminer vers **DATALOG** et les algorithmes d'unification, cadre où les relations avec les classes de complexité sont mieux connues et se ramènent à des questions d'ordonnancement ou non des structures manipulées. Cette approche n'est pas sans rappeler les questions liées à **PURPLE** : sur structure ordonnée, ce langage est trivialement égal à L.

La **LL** a donné les réseaux de preuves, d'interaction, la **GdI**, autant d'outils qui s'éclairent mutuellement, interagissent avec le λ-calcul et les graphes de partage, et sont en rapport avec la complexité. Nous avons contribué à enrichir ces échanges avec la représentation de classes parallèles – au moyen de circuits de preuves – et non-déterministes – avec les entiers Logspace.

Il nous reste bien sûr à renforcer ces approches, en clarifiant les correspondances entre **MTA** et circuits de preuve, en affinant leur critère d'uniformité. Les réseaux de preuve peuvent désormais simuler des circuits booléens

de profondeur constante de façon efficace, et seront bientôt à même – une fois abstraits à l'aide de la **GdI** – d'exprimer des classes non-déterministes, le tout sans recourir exclusivement à la gestion des modalités comme bornes de la complexité. Les graphes de partage permettent eux aussi de contourner la puissance des exponentiels, en reportant naturellement le plus possible la duplication des calculs intermédiaires. Cette pluralité d'approches – qui emploient la logique à la fois comme outil et objet d'étude –, dresse une première topologie de la complexité sous-polynomiale, qui devrait être en mesure d'offrir en retour un cadre unificateur de travail.

Références

[AG98] A. Asperti et S. Guerrini : *The optimal implementation of functional programming languages*, volume 45. Cambridge Univ Pr, 1998.

[Aub11] Clément Aubert : Sublogarithmic uniform boolean proof nets. *In* Jean-Yves Marion, éditeur : *DICE*, volume 75 de *EPTCS*, pages 15–27, 2011.

[BM10] Patrick Baillot et Damiano Mazza : Linear logic by levels and bounded time complexity. *Theoretical Computer Science*, 411(2): 470–503, 2010.

[BM12] Guillaume Bonfante et Virgile Mogbil : A circuit uniformity sharper than "dlogtime". preprint LIPN, 2012.

[BP01] Patrick Baillot et Marco Pedicini : Elementary complexity and geometry of interaction. *Fundamenta Informaticae*, 45(1-2):1–31, 2001.

[Coo85] Stephen A. Cook : A taxinomy of problems with fast parallel algorithms. *Information and Control*, 64(1-3):2–22, 1985.

[CS76] Ashok K. Chandra et Larry J. Stockmeyer : Alternation. *In 17th annual symposium on Foundations of Computer Science*, pages 98–108. IEEE, 1976.

[DJ03] Vincent Danos et Jean-Baptiste Joinet : Linear logic and elementary time. *Information and Computation*, 183(1):123–137, 2003.

[Gir89] Jean-Yves Girard : Geometry of interaction 1 : Interpretation of System F. *Studies in Logic and the Foundations of Mathematics*, 127:221–260, 1989.

[Gir95] Jean-Yves Girard : Light linear logic. *In Logic and computational complexity*, pages 145–176. Springer, 1995.

[Gir11a] Jean-Yves Girard : Geometry of interaction V : logic in the hyperfinite factor. *Theoretical Computer Science*, 412(20):1860–1883, 2011.

[Gir11b] Jean-Yves Girard : Normativity in logic. May 2011.

[Hof03] Martin Hofmann : Linear types and non-size-increasing polynomial time computation. *Information and Computation*, 183(1):57–85, 2003.

[Laf04] Yves Lafont : Soft linear logic and polynomial time. *Theoretical Computer Science*, 318(1):163–180, 2004.

[Lag05] Ugo Dal Lago : The geometry of linear higher-order recursion. *In Logic in Computer Science, 2005. LICS 2005. Proceedings. 20th Annual IEEE Symposium on*, pages 366–375. IEEE, 2005.

[MR07] Virgile Mogbil et Vincent Rahli : Uniform circuits, & Boolean proof nets. *In Proceedings of LFCS'07*, volume 4514 de *Lecture Notes in Computer Science*, pages 401–421. Springer, 2007.

[Ruz81] Walter L. Ruzzo : On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383, 1981.

[Sch07] Ulrich Schöpp : Stratified bounded affine logic for logarithmic space. *In Logic in Computer Science, 2007. LICS 2007. 22nd Annual IEEE Symposium on*, pages 411–420. IEEE, 2007.

[Sei12] Thomas Seiller : Interaction graphs : Multiplicatives. *Annals of Pure and Applied Logic*, 2012.

[SH08] Ulrich Schöpp et Martin Hofmann : Pointer programs and undirected reachability. *In Electronic Colloquium on Computational Complexity (ECCC)*, volume 15, 2008.

[Ter04] Kazushige Terui : Proof Nets and Boolean Circuits. *In Proceedings of LICS'04*, pages 182–191, 2004.

[Vol99] Heribert Vollmer : *Introduction to Circuit Complexity : A Uniform Approach*. Springer Verlag, 1999.