

Logarithmic Space and Permutations*

Clément Aubert and Thomas Seiller

Abstract

In a recent work, Girard [6] proposed a new and innovative approach to computational complexity based on the proofs-as-programs correspondence. The authors recently showed [1, 2] how Girard proposal succeeds in obtaining characterizations of L and co-NL languages as sets of operators in the hyperfinite factor of type II_1 .

1 Introduction

Logic, and more precisely proof theory — the domain whose purpose is the formalization and study of mathematical proofs — recently yielded numerous developments in theoretical computer science. These developments are founded on a correspondence, often called Curry-Howard correspondence, between mathematical proofs and computer programs (usually formalized in lambda-calculus). The main interest of this correspondence lies in its dynamic nature: program execution corresponds to a procedure on mathematical proofs known as the cut-elimination procedure.

Based on a deep study of the formalization of proof in linear logic, in particular their formalization as *proof nets*, Jean-Yves Girard initiated the *geometry of interaction* program. This program, in a first approximation, intends to define semantics of proofs that accounts for the dynamics of the cut-elimination procedure. Through the correspondence between proofs and programs, this would define a semantics of programs that accounts for the dynamics of their execution. Since the introduction of this program, Girard proposed several constructions to realize it, where proofs are interpreted by operators in a von Neumann algebra.

Linear logic has already given birth to a number of developments in the field of Implicit Computational Complexity. In particular through the definition of constrained logical systems — obtained by restricting the rules governing the exponential connectives — which characterize some complexity classes [4]. It was also shown [8] that a characterization of logarithmic space computation can be obtained if one restricts both the rules of exponential connectives and the use of universal quantifiers. Finally, a variation on the notion of linear logic proof nets succeeded in characterizing the classes **NC** of problems that can be efficiently parallelized [11, 3].

*Work partially supported by the ANR projects ANR-2010-BLAN-021301 LOGOI and ANR-08-BLAN-0211-01 COMPLICE.

Recently Jean-Yves Girard proposed [6] a new approach for the study of complexity classes that was inspired by his latest construction of a geometry of interaction [5]. Using the crossed product construction between a von Neumann algebra and a group acting on it, he proposed to characterize complexity classes as sets of operators obtained through the internalization of outer automorphisms of the type II_1 hyperfinite factor. The authors showed in recent papers [1, 2] that this approach succeeds in defining a characterization of the complexity classes **co-NL** and **L** as sets of operators in the type II_1 hyperfinite factor.

2 The Basic Picture

The construction uses an operator-theoretic construction known as the crossed product of an algebra by a group acting on it. In a nutshell, the crossed product construction $\mathfrak{A} \rtimes_\alpha G$ of a von Neumann algebra $\mathfrak{A}$ and a group action $\alpha : G \rightarrow \text{Aut}(\mathfrak{A})$ defines a von Neumann algebra containing $\mathfrak{A}$ and unitaries that internalize the automorphisms $\alpha(g)$ for $g \in G$. For this, one considers the Hilbert space¹ $\mathbb{K} = L^2(G, \mathbb{H})$ where $\mathbb{H}$ is the Hilbert space $\mathfrak{A}$ is acting on, and one defines two families of unitary operators² in $\mathcal{L}(\mathbb{K})$:

- the family $\{\pi(u) \mid u \in \mathfrak{A} \text{ unitary}\}$ which is a representation of $\mathfrak{A}$ on $\mathbb{K}$;
- the family $\{\lambda(g) \mid g \in G\}$ which contains unitaries that internalize the automorphisms $\lambda(g)$.

Then the crossed product $\mathfrak{A} \rtimes_\alpha G$ is the von Neumann algebra generated by these two families.

As a by-product of Girard's work on Geometry of Interaction [5], we know³ how to represent integers as operators in the type II_1 hyperfinite factor $\mathfrak{R}$. In fact, an integer n is represented as a matrix M_n (an operator on a finite-dimensional Hilbert space) corresponding to the axiom links in the proof net which represents n . But the size of M_n depends on the value of n ! In order to deal with this non-uniformity, we use the fact that any matrix algebra can be embedded in the hyperfinite factor $\mathfrak{R}$. A representation of the integer n is thus defined as an embedding of M_n in the hyperfinite factor. Since there are no natural choice of a particular embedding, a given integer does not have a unique representation in $\mathfrak{R}$, but a family of representations. Luckily, any two representations of a given integer are unitarily equivalent (we denote by $\mathfrak{M}_k(\mathbb{C})$ the algebra of $k \times k$ matrices):

¹The construction $L^2(G, \mathbb{H})$ is a generalization of the well-known construction of the Hilbert space of square-summable functions: in case G is considered with the discrete topology, the elements are functions $f : G \rightarrow \mathbb{H}$ such that $\sum_{g \in G} \|f(g)\|^2 < \infty$.

²Recall that in the algebra $\mathcal{L}(\mathbb{H})$ of bounded linear operators on the Hilbert space $\mathbb{H}$ (we denote by $\langle \cdot, \cdot \rangle$ its inner product), there exists an anti-linear involution $(\cdot)^*$ such that for any $\xi, \eta \in \mathbb{H}$ and $A \in \mathcal{L}(\mathbb{H})$, $\langle A\xi, \eta \rangle = \langle \xi, A^*\eta \rangle$. This *adjoint operator* coincides with the conjugate-transpose in the algebras of square matrices. A *unitary operator* u is an operator such that $uu^* = u^*u = 1$.

³A detailed explanation of this representation can be found in the authors' previous work [1] or in the second author's PhD thesis [10].

Proposition 1 ([1, Proposition 9]). *Let $N, N' \in \mathfrak{M}_6(\mathbf{C}) \otimes \mathfrak{R}$ be two representations of a given integer. There exists a unitary $u \in \mathfrak{R}$ such that $N = (1 \otimes u)^* N' (1 \otimes u)$.*

The next step is then to define a representation of machines, or algorithms, as operators that do not discriminate between two distinct representations of a given integer, i.e. operators that act uniformly on the set of representations of a given integer. As the authors shown in their previous work, the crossed product construction allows one to characterize an algebra of operators acting in such a uniform way. We recall the construction in the particular case of the group of finite permutations, which is the only setting studied in this work.

The algebra $\mathfrak{R}$ can be embedded in the infinite tensor product algebra $\mathfrak{T} = \bigotimes_{n \in \mathbf{N}} \mathfrak{R}$ through the morphism $\iota_0 : x \mapsto x \otimes 1 \otimes 1 \otimes \dots$. We will denote by $\mathfrak{N}$ the image of $\mathfrak{R}$ in $\mathfrak{T}$. Notice that this tensor product algebra is isomorphic to $\mathfrak{R}$. The idea of Girard is then to use the action of the group $\mathcal{S}$ of finite permutations of $\mathbf{N}$ onto $\mathfrak{T}$, defined by $\sigma.(x_0 \otimes x_1 \otimes \dots \otimes x_k \otimes \dots) = x_{\sigma^{-1}(0)} \otimes x_{\sigma^{-1}(1)} \otimes \dots \otimes x_{\sigma^{-1}(k)} \otimes \dots$. This group action defines a sub-algebra $\mathfrak{G}$ of the crossed product algebra $\mathfrak{K} = (\bigotimes_{n \in \mathbf{N}} \mathfrak{R}) \rtimes \mathcal{S}$. This algebra $\mathfrak{G}$ is the algebra generated by the family of unitaries $\lambda(\sigma)$ for $\sigma \in \mathcal{S}$, and we think of it as the algebra of machines, or algorithms. As it turns out, the elements of this algebra act uniformly on the set of representations of a given integer:

Proposition 2 ([1, Proposition 11]). *Let $N, N' \in \mathfrak{M}_6(\mathbf{C}) \otimes \mathfrak{T}$ be two representations of a given integer, and $\phi \in \mathfrak{M}_6(\mathbf{C}) \otimes \mathfrak{G}$. Then ϕN is nilpotent if and only if $\phi N'$ is nilpotent.*

From this proposition, one can justify that the following definition makes sense:

Definition 3. Let $\phi \in \mathfrak{M}_6(\mathbf{C}) \otimes \mathfrak{G}$ be an operator. We define the language accepted by ϕ by:

$$[\phi] = \{n \in \mathbf{N} \mid \phi N_n \text{ nilpotent, } N_n \text{ a representation of } n\}$$

We then show that one can define two sets of operators, namely the sets P_+ and $P_{+,1}$ defined below (Definition 7), that characterize **co-NL** and **L**, i.e. such that $[P_+] = \{[\phi] \mid \phi \in P_+\} = \mathbf{co-NL}$ and $[P_{+,1}] = \{[\phi] \mid \phi \in P_{+,1}\} = \mathbf{L}$.

3 Pointers, Operators and Logarithmic Space

We introduced [1] the notion of Non-Deterministic Pointer Machines (NDPM) in order to account for the computational expressivity allowed by the use of operators. A pointer machine is given by a set of pointers that can move back and forth on the input tape and read (but not write) the values it contains, together with a set of states. Given a pointer p_i , only one of three different *instructions* can be performed at each step: p_i+ , i.e. "move one step forward", p_i- , i.e. "move one step backward" and ϵ_i , i.e. "do not move". Let $I_{1,\dots,p} = \{p_i+, p_i-, \epsilon_i \mid i \in \{1, \dots, p\}\}$.

Definition 4. A pointer machine with $p \in \mathbf{N}^*$ pointers is a triple $M = \{Q, \Sigma, \rightarrow\}$ where Q is the set of *states*, $Q = \{\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_e\}$; $\Sigma = \{0, 1, \star\}$ is the *alphabet*; and $\rightarrow \subseteq (\Sigma^p \times Q) \times ((I_{1,\dots,p}^p \times Q) \cup \{\mathbf{accept}, \mathbf{reject}\})$ is the transition relation.

M accepts an entry n written in binary on its circular read-only input tape if after a finite number of transitions, all of its branches reach **accept**⁴. If $\rightarrow$ is functional, M is said to be *deterministic*. We denote by $\text{DPM}(p)$ (resp. $\text{NDPM}(p)$) the set of deterministic (resp. non-deterministic) pointer machines with exactly p pointers.

Those pointer machines turn out to be equivalent to the well-studied Multi-Head Finite Automaton [7]. For $k \in \mathbf{N}^*$, we denote by $2\text{NDFA}(k)$ (resp. $2\text{DFA}(k)$) the set of non-deterministic (resp. deterministic) two-way k -head finite automata.

We will denote by **DPM**, **NDPM**, **2DFA** and **2NDFA** the set of languages decided respectively by the set $\cup_{p \in \mathbf{N}^*} \text{DPM}(p)$, $\cup_{p \in \mathbf{N}^*} \text{NDPM}(p)$, $\cup_{k \in \mathbf{N}^*} 2\text{DFA}(k)$ and $\cup_{k \in \mathbf{N}^*} 2\text{NDFA}(k)$. It is a classical result that $2\text{NDFA} = \mathbf{NL}$ and $2\text{DFA} = \mathbf{L}$.

We show the equivalence by exhibiting two families of constant-depth Boolean circuits⁵ that translate Finite Automata into Pointer Machines and reciprocally.

Theorem 5.

$$\mathbf{DPM} = \mathbf{L} \quad \text{and} \quad \mathbf{NDPM} = \mathbf{co-NL}$$

The pointer machines were defined in order to account for the basic computational steps that can be performed by the operators. This can be formally shown by defining, given a pointer machine M , an operator M^* that will decide the same language as M . It turns out that the operator M^* is a *boolean operator*:

Definition 6. Let $\phi \in \mathcal{M}_6(\mathbf{C}) \otimes \mathcal{G} \otimes \mathcal{M}_k(\mathbf{C})$ be an operator that we will write as a $6k \times 6k$ matrix $(a_{i,j})_{1 \leq i,j \leq 6k}$ with coefficients in $\mathcal{G}$. We say that ϕ is a *boolean operator* if $\forall 1 \leq i,j \leq 6k$, $a_{i,j} = \sum_{m=0}^l \lambda(g_m^{i,j})$ where l is an integer.

If M is a NDPM, then M^* is a boolean operator. But in the particular case of M being deterministic, then M^* is a boolean operator such that⁶ $\|M^*\|_1 \leq 1$. This justifies the following definitions:

Definition 7. We define the following sets of operators:

$$P_+ = \{\phi \mid \phi \text{ is a boolean operator}\} \quad P_{+,1} = \{\phi \mid \phi \in P_+ \text{ and } \|\phi\|_1 \leq 1\}$$

From the different results we already exposed, one can naturally obtain the inclusions $\mathbf{L} \subset [P_{+,1}]$ and $\mathbf{co-NL} \subset [P_+]$. As it turns out, the converse inclusions holds⁷, and we get the main theorem:

Theorem 8.

$$[P_{+,1}] = \mathbf{L} \quad \text{and} \quad [P_+] = \mathbf{co-NL}$$

⁴This particularity will ease the simulation by operators and explains why it is more natural for this device to characterize the complementary of a complexity class.

⁵This insures us that the translation can be performed with sublogarithmic resources.

⁶A precise definition of the norm $\|\cdot\|_1$ is given in our second paper [2].

⁷The proof of these inclusions involves some calculations with the crossed product. To sum up, we show that checking the nilpotency of a product ϕN_n where ϕ is in P_+ and N_n is a representation of an integer is equivalent to checking the nilpotency of a particular matrix. And we show that when $\phi \in P_+$ (resp. $\phi \in P_{+,1}$) the nilpotency of this particular matrix can be checked by a non-deterministic (resp. deterministic) Turing machine using only logarithmic space.

4 Conclusion and Perspectives

This work shows how the approach recently proposed by Girard [6] for studying complexity classes succeeds in characterizing the classes **co-NL** and **L**. As a by-product, we obtain a new characterization of these two complexity classes by means of the notion of pointer machines.

The notion of acceptance (nilpotency) used in this work is closely related [9] to the notion of interaction in Girard's geometry of interaction in the hyperfinite factor (GoI5). This should lead to the possibility of defining types in GoI5 corresponding to complexity classes. The combined tools offered by geometry of interaction and the numerous tools and invariants of the theory of operators may then lead to the obtention of separation results.

We also believe that characterizations of new complexity classes can be obtained, and in particular the classes **P** and **co-NP**. These characterizations may be obtained through the use of a more complex group action in the crossed product construction, or by defining a suitable superset of P_+ .

References

- [1] Clément Aubert and Thomas Seiller. Characterizing co-NL by a group action. *Arxiv preprint arXiv:1209.3422*, 2012.
- [2] Clément Aubert and Thomas Seiller. Logarithmic space and permutations. *in preparation*, 2013.
- [3] Clément Aubert. Sublogarithmic uniform boolean proof nets. In Jean-Yves Marion, editor, *DICE*, volume 75 of *EPTCS*, pages 15–27, 2011.
- [4] Vincent Danos and Jean-Baptiste Joinet. Linear logic & elementary time. *Information and Computation*, 183:2003, 2001.
- [5] Jean-Yves Girard. Geometry of interaction V: Logic in the hyperfinite factor. *Theoretical Computer Science*, 412:1860–1883, 2011.
- [6] Jean-Yves Girard. Normativity in logic. *Epistemology vs. Ontology*, 2011.
- [7] Markus Holzer, Martin Kutrib, and Andreas Malcher. Multi-head finite automata: Characterizations, concepts and open problems. In *CSP*, 2008.
- [8] Ulrich Schöpp. Stratified bounded affine logic for logarithmic space. In *Proceedings of LICS '07*, 2007.
- [9] Thomas Seiller. Interaction graphs: Additives. *Arxiv preprint arXiv:1205.6557*, 2012.
- [10] Thomas Seiller. *Logique dans le facteur hyperfini : géométrie de l'interaction et complexité*. PhD thesis, Université de la Méditerranée, 2012.
- [11] Kazushige Terui. Proof Nets and Boolean Circuits. In *Proceedings of LICS '04*, 2004.