Towards a *Complexity-through-Realizability* Theory

Thomas Seiller

Abstract

We explain how recent developments in the fields of realizability models for linear logic [Sei14e] – or *geometry of interaction* – and implicit computational complexity [AS14, AS15] can lead to a new approach of implicit computational complexity. This semantic-based approach should apply uniformly to various computational paradigms, and enable the use of new mathematical methods and tools to attack problem in computational complexity. This paper provides the background, motivations and perspectives of this *complexity-through-realizability* theory to be developed, and illustrates it with recent results [Sei15].

1 Introduction

Complexity theory lies at the intersection between mathematics and computer science, and studies the amount of resources needed to run a specific program (complexity of an algorithm) or solve a particular problem (complexity of a problem). I will explain how it is possible to build on recent work in *realizability models for linear logic* – a mathematical model of programs and their execution – to provide new characterizations of existing complexity classes. It is hoped that these characterizations will enable new mathematical techniques, tools and invariants from the fields of operators algebras and dynamical systems, providing researchers with new tools and methods to attack long-standing open problems in complexity theory.

The *complexity-through-realizability* theory I propose to develop will provide a unified framework for studying many computational paradigms and their associated computational complexity theory grounded on well-studied mathematical concepts. This should provide a good candidate for a theory of complexity for computational paradigms currently lacking an established theory (e.g. concurrent processes), as well as contribute to establish a unified and well-grounded account of complexity for higher-order functionals.

Even though it has been an established discipline for more than 50 years [HS65b], many questions in complexity theory, even basic ones, remain open. During the last twenty years, researchers have developed new approaches based on logic: they offer solid, machine-independent, foundations and provide new tools and methods. Amongst these approaches, the fields of Descriptive Complexity (DC) and Implicit Computational Complexity (ICC) lead to a number of new characterizations of complexity classes. These works laid grounds for both theoretical results [Imm88] and applications such as typing systems for complexity

constrained programs and type inference algorithms for statically determining complexity bounds.

The *complexity-through-realizability* theory I propose to develop is related to those established logic-based approaches. As such, it inherits their strengths: it is machine-independent, provides tools and methods from logic and gives grounds for the above mentioned applications. Furthermore, it builds on state-of-the-art theoretical results on realizability models for linear logic [Sei14e] using well-studied mathematical concepts from operators algebras and dynamical systems. As a consequence, it opens the way to use against complexity theory's open problems the many techniques, tools and invariants that were developed in these disciplines.

We illustrate the approach by explaining how first results were recently obtained by capturing a large family of complexity classes corresponding to various notions of automata. Indeed, we provided [Sei15] realizability models in which types of binary predicates correspond to the classes of languages accepted by one-way (resp. two-way) deterministic (resp. non-deterministic, resp. probabilistic) multi-head automata. This large family of languages contains in particular the classes REG (regular languages), STOC (stochastic languages), L (logarithmic space), NL (non-deterministic logarithmic space), coNL (complementaries of languages in NL), and PL (probabilistic logarithmic space).

2 Background

2.1 Complexity Theory

Complexity theory is concerned with the study of how many resources are needed to perform a specific computation or to solve a given problem. The study of *complexity classes* – sets of problems which need a comparable amount of resources to be solved, lies at the intersection of mathematics and computer science. Although a very active and established field for more than fifty years [Cob65, Coo71, HS65a, Sav70], a number of basic problems remain open, for instance the famous "millennium problem" of whether P equals NP or the less publicized but equally important question of whether P equals L. In recent years, several results have greatly modified the landscape of complexity theory by showing that proofs of separation (i.e. inequality) of complexity classes are hard to come by, pointing out the need to develop new theoretical methods. The most celebrated result in this direction [RR97] defines a notion of *natural proof* comprising all previously developed proof methods and shows that no "natural proof" can succeed in proving separation.

Mathematicians have then tried to give characterizations of complexity classes that differ from the original machine-bound definitions, hoping to enable methods from radically different areas of mathematics. Efforts in this direction lead to the development of Descriptive Complexity (DC), a field which studies the types of logics whose individual sentences characterize exactly particular complexity classes. Early developments were the 1974 Fagin-Jones-Selman results [Fag74, JS74] characterizing the classes NEXP and NP. Many such characterizations have then been given [CG96, DG08, GG95] and the method led Immerman to a proof of the celebrated Immerman-Szelepcsényi theorem [Imm88, Sze87] stating the two complexity classes coNL and NL are equal (though Szelepc-

sényi's proof does not use logic-based methods).

Implicit Computational Complexity (ICC) develops a related approach whose aim is to study algorithmic complexity only in terms of restrictions of languages and computational principles. It has been established since Bellantoni and Cook' landmark paper [BC92], and following work by Leivant and Marion [LM93, LM94]. Amongst the different approaches to ICC, several results were obtained by considering syntactic restrictions of *linear logic* [Gir87], a refinement of intuitionnistic logic which accounts for the notion of resources. Linear logic introduces a modality ! marking the "possibility of duplicating" a formula A: the formula A shall be used exactly once, while the formula $!A$ can be used any number of times. Modifying the rules governing this modality then yields variants of linear logic having computational interest: this is how constrained linear logic systems, for instance BLL [GSS92] and ELL [DJ03], are obtained. However, only a limited number of complexity classes were characterized in this way, and the method seems to be limited by its syntactic aspect: while it is easy to modify existing rules, it is much harder to find new, alternative, rules from scratch. The approach taken in my research project does not suffer from these limitations, allowing for subtle distinctions unavailable to the syntactic techniques of ICC.

2.2 Realizability Models for Linear Logic and Complexity

Concurrently to these developments in computational complexity, and motivated by disjoint questions and interests, Girard initiated the Geometry of Interaction (GoI) program [Gir89b]. This research program aims at obtaining particular kinds of *realizability* models (called GoI models) for linear logic. Realizability was first introduced [Kle45] as a way of making the Brouwer-Heyting-Kolmogorov interpretation of constructivism and intuitionistic mathematics precise; the techniques were then extended to classical logic, for instance by Krivine [Kri01], and linear logic. The GoI program naturally and quickly arose as a well-suited tool for the study of computational complexity. Using the first GoI model [Gir89a], Abadi, Gonthier and Lévy [GAL92] showed the optimality of Lamping's reduction in lambda-calculus [Lam90]. It was also applied in implicit computational complexity [BP01], and was the main inspiration behind dal Lago's context semantics [Lag09]. On a more practical side, let us mention the *Geometry of Synthesis* program initiated by Ghica [Ghi07, GS10, GS11, GSS11]. This program, inspired by geometry of interaction, aims at obtaining logical synthesis methods for VLSI (Very Large Systems Integration) designs.

More recently the geometry of interaction program inspired a new approach to implicit computational complexity. These new methods were initiated by Girard [Gir12] and have known a rapid development. They lead to a series of results in the form of new characterizations of the classes coNL [Gir12, AS14], L [AS15, ABPS14] and P [ABS15]. Unfortunately, although the construction of realizability models and the characterizations of classes are founded on similar techniques, they are two distinct, unrelated, constructions. The approach I propose to develop in my research project will in particular bridge this gap and provide similar characterizations which will moreover allow the use of both logical and realizability-specific methods.

3 A *Complexity-through-Realizability* Theory

3.1 Technical Background and Motivations

About ten years ago, Girard showed [Gir06] that the restriction to the unit ball of a von Neumann algebra of the so-called "feedback equation", which represents the execution of programs in GoI models, always has a solution. Moreover, previous and subsequent work showed the obtained GoI model interprets, depending on the choice of the von Neumann algebra, either full linear logic [Gir95] or the constrained system ELL which characterizes elementary time computable functions [Sei12b]. This naturally lead to the informal conjecture that there should be a correspondence between von Neumann algebras and complexity constraints.

This deep and promising idea turned out to be slightly inexact and seemingly difficult to exploit. Indeed, I showed [Sei12b, Sei14a] that the expressivity of the logic interpreted in the model depends not only on the enveloping algebra $\mathfrak{N}$ but also on a maximal abelian sub-algebra (masa) $\mathfrak{A}$ of $\mathfrak{N}$, hinting at a refined conjecture stating that complexity constraints correspond to such couples $(\mathfrak{A},\mathfrak{N})$. This approach is however difficult to extend to other constrained logical systems for two reasons. The first reason is that the theory of maximal abelian sub-algebras in von Neumann algebras is an involved subject matter still containing large numbers of basic but difficult open problems [SS08]. The second is that even though some results were obtained, no intuitions were gained about what makes the correspondence between couples $(\mathfrak{A},\mathfrak{N})$ and complexity constraints work.

Some very recent work of mine provides the foundational grounds for a new, tractable way of exploring the latter refined conjecture. This series of work [Sei12a, Sei14b, Sei14e, Sei14d] describes a general systematic construction of realizability models for linear logic which unifies and extends all previous works on the subject, encompassing the last forty years of research in the area. The construction is built upon a generalization of graphs, named *graphings* [Lev95, Gab00], which can be understood either as *geometric realizations* of graphs on a measure space $(X,\mathscr{B},\mu)$, as measurable families of graphs, or as generalized measured dynamical system. It is parametrized by two monoids describing the model of computation and a map describing the realizability structure:

- a monoid Ω used to associate weights to edges of the graphs;
- a measurable map $m : \Omega \to \bar{\mathbf{R}}_{\geqslant 0}$, defining *orthogonality* – accounting for linear negation;
- a monoid $\mathfrak{m}$ – the *microcosm* – of measurable maps from $(X,\mathscr{B},\mu)$ to itself.

A Ω-*weighted graphing in* $\mathfrak{m}$ is then defined as a directed graph F whose edges are weighted by elements in Ω, whose vertices are measurable subsets of the measurable space $(X,\mathscr{B})$, and whose edges are *realized* by elements of $\mathfrak{m}$, i.e. for each edge e there exists an element ϕ_e in $\mathfrak{m}$ such that $\phi_e(s(e)) = t(e)$, where s,t denote the source and target maps. Based on this notion, and an orthogonality relation defined from the map m, I obtained a systematic method for constructing realizability models for linear logic summarized in the following theorem.

Theorem 1 (Seiller [Sei14e])**.** *Let Ω be a monoid, $\mathfrak{m}$ a microcosm and $m : \Omega \to \bar{\mathbf{R}}_{\geqslant 0}$ a measurable map. There exists a deterministic (resp. probabilistic, resp. non-deterministic) GoI model of multiplicative-additive linear logic whose objects are Ω-weighted deterministic (resp. probabilistic, resp. non-deterministic) graphings in $\mathfrak{m}$ and whose orthogonality depends on m.*

Let us notice that a microcosm $\mathfrak{m}$ generates a measurable equivalence relation which, by the Feldman-Moore construction [FM77], induces a couple $(\mathfrak{A}, \mathfrak{N})$ where $\mathfrak{A}$ is a maximal abelian subalgebra of the von Neumann algebra $\mathfrak{N}$. This theorem thus generalizes Girard's result since it shows that a microcosm (identified with a couple $(\mathfrak{A}, \mathfrak{N})$) induces *two* models of computation: a deterministic model (the equivalent of Girard's unit ball restriction) and a non-deterministic model (not available to Girard techniques because of divergence issues). It shows in fact much more as it exhibits an infinite family of structures of realizability models (parametrized by the map $m : \Omega \to \mathbf{R}_{\geqslant 0} \cup \{\infty\}$) on any model obtained from a microcosm. This extends drastically Girard's approach for which only two such structures were defined until now: an "orthogonality-as-nilpotency" structure in the algebra $\mathscr{L}(\mathbb{H})$ [Gir89a] and another one defined using the Fuglede-Kadison determinant [FK52] in the type II_1 hyperfinite factor [Gir11].

3.2 Methodology

We can now explain the proposed methodology for defining a *Complexity-through-Realizability* theory.

The notion of Ω-weighted $\mathfrak{m}$-graphings for given monoid Ω and microcosm $\mathfrak{m}$ yields a very general yet tractable mathematical notion of algorithm, as the microcosm $\mathfrak{m}$ can naturally be understood as a set of computational principles [Sei14c]. It therefore provides an interesting middle-ground between usual computational models, for instance automata, and mathematical techniques from operator algebras and dynamical systems. It comprises Danos' interpretation of pure lambda-calculus (a Turing complete model of computation) in terms of operators [Dan90], but it is not restricted to sequential algorithms as it will be shown to provide an interpretation of probabilistic and quantum programs. It also provides characterizations of usual complexity classes as types of predicates over binary words $\mathbf{Words}_{\Sigma}^{(2)} \Rightarrow \mathbf{Bool}$, which will lead to a partial proof of the above conjecture by showing a correspondence between families of microcosms and complexity constraints.

Work in this direction will establish these definitions of algorithms and complexity constraints as a uniform, homogeneous, machine-independent approach to complexity theory. The methods developed in this setting, either adapted from DC/ICC or new, will apply to probabilistic/quantum complexity classes as much as sequential classes. In particular, it will offer a framework where comparison between non-classical and classical classes can be performed. It will also expand to computational paradigms where no established theory of complexity exists, providing a strong and coherent proposition for such.

It will extend the approach of ICC and DC as it will go beyond the syntactical restrictions they are suffering from. In particular, it will provide a new method for defining logical systems corresponding to complexity classes: the realizability model construction gives a systematic way to define a logic corresponding to the underlying computational model. It will also extend the GoI model approach to complexity by reconciling the logical and complexity aspects, allowing the use of both logical and realizability-specific methods.

Lastly, the approach I propose to develop does not naturally fall into the usual pitfalls for the obtention of separation results. Therefore, it provides a framework which will potentially offer separation methods, e.g. using invariants for the well-established mathematical notions it is founded upon.

4 First Results

In this section, I expose some recent results obtained by applying the methodology described above [Sei15]. We obtain in this way a number of characterizations of complexity classes, among which REG, STOC, L, NL, coNL and PL.

4.1 Graphings

Definition 2. Let $(X, \mathscr{B}, \lambda)$ be a measured space. We denote by $\mathscr{M}(X)$ the set of non-singular transformations[1] $X \to X$. A *microcosm* of the measured space X is a subset $\mathfrak{m}$ of $\mathscr{M}(X)$ which is closed under composition and contains the identity.

In the following, we will consider a notion of graphing depending on a *weight-monoid* Ω, i.e. a monoid $(\Omega, \cdot, 1)$ which contains the possible weights of the edges.

Definition 3 (Graphings)**.** Let $\mathfrak{m}$ be a microcosm of a measured space $(X, \mathscr{B}, \lambda)$ and V^F a measurable subset of X. A Ω-*weighted graphing in* $\mathfrak{m}$ of carrier V^F is a countable family $F = \{(\omega_e^F, \phi_e^F : S_e^F \to T_e^F\}_{e \in E^F}$, where, for all $e \in E^F$ (the set of *edges*):

- ω_e^F is an element of Ω, the *weight* of the edge e;
- $S_e^F \subset V^F$ is a measurable set, the *source* of the edge e;
- $T_e^F = \phi_e^F(S_e^F) \subset V^F$ is a measurable set, the *target* of the edge e;
- ϕ_e^F is the restriction of an element of $\mathfrak{m}$ to S_e^F, the *realization* of the edge e.

I showed in earlier work [Sei14b] how one can construct models of multiplicative-additive linear logic where proofs are interpreted as graphs. This construction relied on a single property, called the *trefoil property*, which relates two simple notions:

- the *execution* $F :: G$ of two graphs, a graph defined as a set of paths;
- the *measurement* $[\![F, G]\!]_m$, a real number computed from a set of cycles.

These constructions can be extended to the more general framework where proofs are interpreted as graphings. Indeed, the notions of paths and cycles in a graphings are quite natural, and from two graphings F, G in a microcosm $\mathfrak{m}$ one can define its execution $F :: G$ which is again a graphing in $\mathfrak{m}$[2]. A more involved argument then shows that the trefoil property holds for a family of measurements $[\![\cdot, \cdot]\!]_m$, where $m : \Omega \to \mathbf{R}_{\geqslant 0} \cup \{\infty\}$ is any measurable map. These results are obtained as a generalization of constructions considered in my PhD thesis[3] [Sei12b].

Theorem 4 (Non-deterministic model)**.** *Let Ω be a monoid and $\mathfrak{m}$ a microcosm. The set of Ω-weighted graphings in $\mathfrak{m}$ yields a model, denoted by $\mathbb{M}[\Omega, \mathfrak{m}]$, of multiplicative-additive linear logic.*

In most of the models, one can define some exponential connectives. In particular, all models considered later on have the necessary structure to define an exponential modality !. Let us notice however that the notion of exponential

[1] A non-singular transformation $f : X \to X$ is a measurable map which preserves the sets of null measure, i.e. $\lambda(f(A)) = 0$ if and only if $\lambda(A) = 0$.

[2] As a consequence, a microcosm is a closed world for the execution of programs.

[3] In the cited work, the results were stated in the particular case of the microcosm of measure-preserving maps on the real line.

modality we are considering here is extremely weak, as most models won't validate the functorial promotion rule. The only rule that is assured to be satisfied by the exponential connectives we will consider is the contraction rule, i.e. for any type $\mathbf{A}$, one has $!\mathbf{A} \multimap !\mathbf{A} \otimes !\mathbf{A}$. These very weak exponential connectives will turn out to be of great interest: we obtain in this way models of linear logic where the exponentials are weaker than what is obtained through syntactic consideration in systems like BLL, SLL, etc. and characterize low complexity classes.

4.2 Models of Computation

Before explaining how one can characterize complexity classes in this way, we first state refinements of the previous theorem. We first define the notion of *deterministic graphing*.

Definition 5 (Deterministic graphings)**.** A Ω-weighted graphing G is *deterministic* when:

- for all $e \in E^G$, $\omega_e^G \in \{0,1\}$;
- the following set is of null measure:

$$\{x \in \mathbf{X} \mid \exists e \neq e' \in E^G, x \in S_e^G \cap S_{e'}^G\}$$

If the graphing G satisfies only the first item, we will say that G is a *non-deterministic graphing*.

We then prove that the notions of deterministic and non-deterministic graphings are closed under composition, i.e. if F, G are deterministic graphings, then their execution $F :: G$ is again a deterministic graphing. This shows that the sets of deterministic and non-deterministic graphings define submodels of $\mathbb{M}[\Omega, \mathfrak{m}]$.

Theorem 6 (Deterministic model)**.** *Let Ω be a monoid and $\mathfrak{m}$ a microcosm. The set of Ω-weighted* deterministic *graphings in $\mathfrak{m}$ yields a model, denoted by $\mathbb{M}_m^{\mathrm{det}}[\Omega, \mathfrak{m}]$, of multiplicative-additive linear logic. The set of Ω-weighted* non-deterministic *graphings in $\mathfrak{m}$ yields a model, denoted by $\mathbb{M}_m^{\mathrm{ndet}}[\Omega, \mathfrak{m}]$, of multiplicative-additive linear logic*

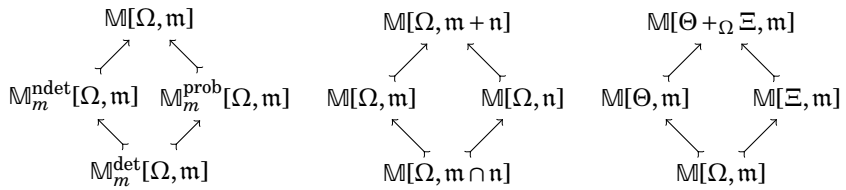
One can also consider several other classes of graphings. We explain here the simplest non-classical model one could consider, namely that of *probabilistic graphings*. In order for this notion to be of real interest, one should suppose that the unit interval $[0,1]$ endowed with multiplication is a submonoid of Ω.

Definition 7 (Probabilistic graphings)**.** A Ω-weighted graphing G is *probabilistic* when:

- for all $e \in E^G$, $\omega_e^G \in [0,1]$;
- the following set is of null measure:

$$\{x \in \mathbf{X} \mid \sum_{e \in E^G, x \in S_e^G} \omega_e^G \neq 1\}$$

It turns out that this notion of graphing also behaves well under composition, i.e. there exists a *probabilistic* submodel of $\mathbb{M}[\Omega, \mathfrak{m}]$, namely the model of *probabilistic graphings*.



(a) Different type of graphings (b) Different microcosms (c) Different weight monoids

Figure 1: Inclusions of models

Theorem 8 (Probabilistic model)**.** *Let Ω be a monoid and $\mathfrak{m}$ a microcosm. The set of Ω-weighted* probabilistic *graphings in $\mathfrak{m}$ yields a model, denoted by $\mathbb{M}_m^{\mathrm{prob}}[\Omega,\mathfrak{m}]$, of multiplicative-additive linear logic.*

These models are all submodels of the single model $\mathbb{M}[\Omega,\mathfrak{m}]$. Moreover, other inclusions of models can be obtained by playing on the other parameters, namely the weight monoid Ω and the microcosm $\mathfrak{m}$. For instance, given two microcosms $\mathfrak{m}\subset\mathfrak{n}$, it is clear that a graphing in $\mathfrak{m}$ is in particular a graphing in $\mathfrak{n}$. This inclusion actually extends to an embedding of the model $\mathbb{M}[\Omega,\mathfrak{m}]$ into $\mathbb{M}[\Omega,\mathfrak{n}]$ which preserves most logical operations[4]. Moreover, given two microcosms $\mathfrak{m}$ and $\mathfrak{n}$, one can define the *smallest common extension* $\mathfrak{m}+\mathfrak{n}$ as the compositional closure of the set $\mathfrak{m}\cup\mathfrak{n}$. The model $\mathbb{M}[\Omega,\mathfrak{m}+\mathfrak{n}]$ then contains both models $\mathbb{M}[\Omega,\mathfrak{m}]$ and $\mathbb{M}[\Omega,\mathfrak{n}]$ through the embedding just mentioned. In the same way, an inclusion of monoids $\Omega\subset\Gamma$ yields an embedding of the the model $\mathbb{M}[\Omega,\mathfrak{m}]$ into $\mathbb{M}[\Gamma,\mathfrak{m}]$. For instance, the model $\mathbb{M}[\{1\},\mathfrak{m}]$ is a submodel of $\mathbb{M}[\Omega,\mathfrak{m}]$ for any monoid Ω. One can also define, given weight monoids Ω, Θ and Ξ with monomorphisms $\Omega\to\Theta$ and $\Omega\to\Xi$, the model $\mathbb{M}[\Theta+_\Omega\Xi,\mathfrak{m}]$ where $\Theta+_\Omega\Xi$ denotes the amalgamated sum of the monoids. Figure 1 illustrates some of these inclusions of models.

We will now explain how these models can yields characterizations of several complexity classes. Before going into details about these characterizations, let us define a number of complexity classes – all of them definable by classes of automata.

Definition 9. For each integer i, we define:

- the class 2DFA(i) (resp. 1DFA(i)) as the set of languages accepted by deterministic two-way (resp. one-way) multihead automata with at most i heads;
- the class 2NFA(i) (resp. 1NFA(i)) as the set of languages accepted by two-way (resp. one-way) multihead automata with at most i heads;
- the class CO2NFA(i) (resp. CO1NFA(i)) as the set of languages whose complementary language is accepted by two-way (resp. one-way) multihead automata with at most i heads;
- the class 2PFA(i) (resp. 1PFA(i)) as the set of languages accepted by two-way (resp. one-way) probabilistic multihead automata with at most i heads;

[4]It preserves all connectives except for negation.

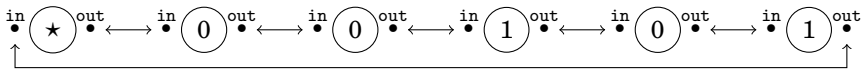


Figure 2: Representation of the word $\mathfrak{w} = 00101$

We also denote by L (resp. P) the class of predicates over binary words that are recognized by a Turing machine using logarithmic space (resp. polynomial time), by NL (resp. NP) its non-deterministic analogue, by coNL (resp. coNP) the set of languages whose complementary language lies in L (resp. P). We also denote by PL the class of predicates over binary words that are recognized by a probabilistic Turing machine with unbounded error using logarithmic space.

We don't recall the usual definitions of these variants of multihead automata, which can be easily found in the literature. We simply recall some classical results:

$$\cup_{i\in\mathbf{N}} 2\mathrm{DFA(i)} = \mathrm{L} \qquad \cup_{i\in\mathbf{N}} 2\mathrm{NFA(i)} = \mathrm{NL} \qquad \cup_{i\in\mathbf{N}} 2\mathrm{PFA(i)} = \mathrm{PL}$$

4.3 From Regular Languages to Logarithmic Space

In the models of linear logic we described, one can easily define the type $\mathbf{Words}^{(2)}_{\Sigma}$ of words over an arbitrary finite alphabet Σ. The definition of the representation of these binary words comes from the encoding of binary lists in lambda-calculus and is explained thoroughly in previous papers [AS14, AS15]. We won't give the formal definition of what is a representation of a word $\mathfrak{w}$ here, but let us sketch the main ideas. Given a word, say the binary word $\mathfrak{w} = 00101$, we introduce a symbol $\star$ that can be understood as a left-hand end-of-tape marker and consider the list of symbols $\star 00101$. Then, the graphing that will represent $\mathfrak{w}$ is obtained by realizing the directed graph over the set of vertices $\{\star, 0, 1\} \times \{\mathtt{in}, \mathtt{out}\}$ whose edges link the symbols of the list together, i.e. the graph pictured in Figure 2.

We are actually interested in the elements of the type $!\mathbf{Words}^{(2)}_{\Sigma}$. For each word $\mathfrak{w}$, there exists an element $!L_{\mathfrak{w}}$ in the type $!\mathbf{Words}^{(2)}_{\{0,1\}}$ which represents it. We say that a graphing – or *program* – P of type $!\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{Bool}$ *accepts* the word $\mathfrak{w}$ when the execution $P :: W_{\mathfrak{w}}$ is equal to the distinguished element $\mathtt{true} \in \mathbf{Bool}$. The *language* accepted by such a program P is then defined as $[P] = \{\mathfrak{w} \in \mathbf{Words}^{(2)}_{\{0,1\}} \mid \phi :: W_{\mathfrak{w}} = \mathtt{true}\}$.

Definition 10 (Characterization)**.** Let Ω be a monoid, $\mathfrak{m}$ a microcosm and $\mathscr{L}$ a set of languages. We say the type $!\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{Bool}$ *characterizes the set* $\mathscr{L}$ if the following set is equal to $\mathscr{L}$

$$\{[F] \mid F \in !\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{Bool}\}$$

We now consider the measured space $\mathbf{N} \times [0,1]^{\mathbf{N}}$ endowed with the product of the counting measure on $\mathbf{N}$ and the Lebesgue measure on the Hilbert cube $[0,1]^{\mathbf{N}}$. We define the following microcosms:

- $\mathfrak{m}_1$ is the monoid of translations $T_k : (n,x) \mapsto (n+k,x)$;
- $\mathfrak{m}_{i+1}$ is the monoid $\mathfrak{m}_i + \mathfrak{s}_{i+1}$ where $\mathfrak{s}_{i+1}$ is the monoid generated by the single map:

$$s_{i+1} : (n,(x_1,x_2,\ldots)) \mapsto (n,(x_{i+1},x_2,\ldots,x_i,x_1,x_{i+2},\ldots))$$

- $\mathfrak{m}_\infty = \cup_{i \in \mathbf{N}} \mathfrak{m}_i$.

The intuition is that a microcosm $\mathfrak{m}$ represents the set of computational principles available to write programs in the model. The operation + thus extends the set of principles at disposal, increasing expressivity. As a consequence, the set of languages characterized by the type $!\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{Bool}$ becomes larger and larger as we consider extensions of the microcosms. As an example, the microcosm $\mathfrak{m}_1$ corresponds to allowing oneself to compute with automata. Expanding this microcosm by adding a map s_2 yields $\mathfrak{m}_2 = \mathfrak{m}_1 + \mathfrak{s}_2$ and corresponds to the addition of a new computational principle: using a second head.

Theorem 11. *In the model* $\mathbb{M}^{\mathrm{det}}_m[\{1\}, \mathfrak{m}_i]$*, the type* $!\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{Bool}$ *characterizes the class* 2DFA(i).

In particular, the type $!\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{Bool}$ in the model $\mathbb{M}^{\mathrm{det}}_m[\{1\}, \mathfrak{m}_1]$ characterizes the class REG of Regular languages.

Theorem 12. *In the model* $\mathbb{M}^{\mathrm{det}}_m[\{1\}, \mathfrak{m}_\infty]$*, the type* $!\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{Bool}$ *characterizes the class* L.

4.4 Non-deterministic and Probabilistic Computation

All the preceding results have non-deterministic analogues; we consider in this section the model of non-deterministic graphings. To obtain the same types of results in that case, two issues should be dealt with. First one needs to consider programs of a different type since the result of a non-deterministic computation yield sets of booleans and not a single boolean. Thus, programs will be considered as elements of a more general type than in the deterministic case: $!\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{NBool}$, where **NBool** is a specific type definable in the models, somehow a non-deterministic version of the booleans.

The second issue concerns acceptance. While it seemed natural in the deterministic case to ask the computation to yield the element $\mathtt{true} \in \mathbf{Bool}$, one has a choice now. Should one define acceptance as producing at least one element `true` or as producing no element `false`? Both conditions should be considered. In order to obtain a quite general notion of acceptance that can not only capture both notions explained above but extend to other computational paradigms such as probabilistic computation, we use the structure of the realizability models we are working with to define a notion of *test*. Indeed, the models are constructed around an orthogonality relation $\perp$: a test will be an element (or more generally a family of elements) $\mathscr{T}$ of the model and a program P accepts the word $\mathfrak{w}$ if the execution $P :: !L_{\mathfrak{w}}$ is orthogonal to $\mathscr{T}$.

One can then define the language $[M]_{\mathscr{T}}$ as the set of all words $\mathfrak{w}$ that are accepted by M w.r.t. the test $\mathscr{T}$:

$$[M]_{\mathscr{T}} = \{\mathfrak{w} \mid M :: !L_{\mathfrak{w}} \perp \mathscr{T}\}$$

Definition 13. Let Ω be a monoid, $\mathfrak{m}$ a microcosm, $\mathscr{T}$ a test and $\mathscr{L}$ a set of languages. We say the type $!\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{NBool}$ *characterizes the set* $\mathscr{L}$ *w.r.t. the test* $\mathscr{T}$ if the following set is equal to $\mathscr{L}$

$$\{[F]_{\mathscr{T}} \mid F \in !\mathbf{Words}^{(2)}_{\{0,1\}} \multimap \mathbf{NBool}\}$$

In particular, one can show the existence of two tests $\mathscr{T}^{\mathrm{n}}$ and $\mathscr{T}^{\mathrm{co}}$ that correspond to the two notions of acceptance mentioned above and which allows for the characterization of usual non-deterministic classes.

Theorem 14. *In the model* $\mathbb{M}_m^{\mathrm{ndet}}[\{1\}, \mathfrak{m}_{\mathrm{i}}]$*, the type* $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{NBool}$ *characterizes the class* 2NFA(i) w.r.t. the test $\mathscr{T}^{\mathrm{n}}$ and the class CO2NFA(i) w.r.t. the test $\mathscr{T}^{\mathrm{co}}$.

In particular, the type $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{NBool}$ in the model $\mathbb{M}_m^{\mathrm{ndet}}[\{1\}, \mathfrak{m}_1]$ characterizes the class REG of Regular languages.

Theorem 15. *In the model* $\mathbb{M}_m^{\mathrm{ndet}}[\{1\}, \mathfrak{m}_\infty]$*, the type* $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{NBool}$ *characterizes the class* NL w.r.t. the test $\mathscr{T}^{\mathrm{n}}$ and the class CONL w.r.t. the test $\mathscr{T}^{\mathrm{co}}$.

In the case of probabilistic graphings, one can show the existence of a test $\mathscr{T}^{\mathrm{p}}$ which allows for the characterization of probabilistic computation with unbounded error. This leads to the following theorems.

Theorem 16. *In the model* $\mathbb{M}_m^{\mathrm{prob}}[[0,1], \mathfrak{m}_{\mathrm{i}}]$*, the type* $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{NBool}$ *characterizes the class* 2PFA(i) w.r.t. the tests $\mathscr{T}^{\mathrm{p}}$.

In particular, the type $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{NBool}$ in the model $\mathbb{M}_m^{\mathrm{prob}}[[0,1], \mathfrak{m}_1]$ characterizes the class STOC of Stochastic languages.

Theorem 17. *In the model* $\mathbb{M}_m^{\mathrm{prob}}[[0,1], \mathfrak{t} + \mathfrak{p}]$*, the type* $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{NBool}$ *characterizes the class* PL w.r.t. the test $\mathscr{T}^{\mathrm{p}}$.

4.5 And Then More

All of these results are based upon the fact that elements of the type $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{NBool}$ correspond to some kinds of two-way multihead automata, either deterministic, non-deterministic or probabilistic. Several other results can be obtained by modifying the notion of automata considered in three different ways.

The first modification is actually a restriction, that is: can we represent computation by *one-way automata*? One can already answer positively to this question, as the two-way capability of the automata does not really find its source in the programs P in $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{NBool}$ but in the representation of words. One can define an alternative representation of words over an alphabet Σ and a corresponding type $\mathbf{Words}_{\Sigma}^{(1)}$. We then obtain the following results, i.e. the one-way analogue of Theorem 11, Theorem 12, Theorem 14, Theorem 12, Theorem 16 and Theorem 17.

Theorem 18 (One-way characterizations - deterministic case)**.** *In* $\mathbb{M}_m^{\mathrm{det}}[[0,1], \mathfrak{m}_{\mathrm{i}}]$ *(resp.* $\mathbb{M}_m^{\mathrm{det}}[[0,1], \mathfrak{m}_\infty]$*), the type* $!\mathbf{Words}_{\{0,1\}}^{(1)} \multimap \mathbf{Bool}$ *characterizes the class* 1DFA(i) (resp. $\cup_{i \geqslant 1}$1DFA(i)).

Theorem 19 (One-way characterizations - non-deterministic case)**.** *In* $\mathbb{M}_m^{\mathrm{ndet}}[[0,1], \mathfrak{m}_{\mathrm{i}}]$ *(resp.* $\mathbb{M}_m^{\mathrm{ndet}}[[0,1], \mathfrak{m}_\infty]$*), the type* $!\mathbf{Words}_{\{0,1\}}^{(1)} \multimap \mathbf{NBool}$ *characterizes the class* 1NFA(i) (resp. $\cup_{i \geqslant 1}$1NFA(i)) w.r.t. the test $\mathscr{T}^{\mathrm{n}}$ and CO1NFA(i) (resp. $\cup_{i \geqslant 1}$CO1NFA(i)) w.r.t. the test $\mathscr{T}^{\mathrm{co}}$.

Theorem 20 (One-way characterizations - probabilistic case)**.** *In* $\mathbb{M}_m^{\mathrm{prob}}[[0,1], \mathfrak{m}_{\mathrm{i}}]$ *(resp.* $\mathbb{M}_m^{\mathrm{prob}}[[0,1], \mathfrak{m}_\infty]$*), the type* $!\mathbf{Words}_{\{0,1\}}^{(1)} \multimap \mathbf{NBool}$ *characterizes the class* 1PFA(i) (resp. $\cup_{i \geqslant 1}$1PFA(i)) w.r.t. the test $\mathscr{T}^{\mathrm{p}}$.

The second modification is the extension of automata with a *pushdown stack*. Work in this direction has recently lead to a characterization of P in a more syntactical setting [ABS15]. Even though the syntactical approach just mentioned could very well be transported to our setting (it was shown [Sei14e] that elements of the *resolution algebra* can be represented as graphings), this would lead to a characterization based on a microcosm containing non-measure-preserving maps. Even though non-measure-preserving maps are allowed in the general setting of graphings [Sei14e], the use of measure-preserving microcosm is more interesting in view of the possible use of mathematical invariants such as ℓ^2-Betti numbers discussed in the next section. One can find such a measure-preserving microcosm $\mathfrak{p}$ which leads to a characterization of P in both the model of deterministic graphings and the model of non-deterministic graphings because non-determinism do not add any expressivity to the model of deterministic two-way multihead automata with a pushdown stack.

The last modification is the consideration of *quantum graphings*, i.e. graphings computing with complex numbers. This is still a work in progress, but I believe that one can define variants of quantum graphings corresponding to measure-once or measure-many quantum automata, leading to several other characterizations.

5 Perspectives

We proposed here to develop a theory of *complexity-through-realizability*, founded on alternative definitions of the notions of algorithms and complexity classes. This is illustrated by first results showing how a large family of complexity classes (predicates) can be characterized by these techniques. These results are a first step towards a demonstration that these definitions capture and generalize standard ones, offering a unified homogeneous framework for the study of complexity classes.

The *complexity-through-realizability* theory to be developed will therefore have two main objectives. We will first aim at establishing that this new approach to complexity captures, generalizes and extends the techniques developed by previous approaches such as DC and ICC. We will also investigate how the new methods and techniques derived from the mathematical foundations of our approach can be used to address open problems in complexity.

I propose the following three goals to deal with the first objective of the project: obtaining foundational results which will establish the *complexity-through-realizability* approach as an independent research field. Those correspond to three natural sequential steps: show the *complexity-through-realizability* techniques (i) are coherent with classical theory, (ii) generalize and improve state-of-the-art techniques, and finally (iii) provide the first homogeneous theory of complexity for several computational paradigms.

(i) *Show that* Complexity-through-realizability *techniques yield characterizations of usual complexity classes.*

The results presented in this paper are a first step, but they lean on previously known characterizations of complexity classes (predicates) by means of all kinds of automata. An adaptation of work by Asperti and Roversi [AR02] and Baillot [Bai11] should allow for encoding Turing machines in

the realizability models we consider and should lead to characterizations of several other complexity classes (predicates), such as the exponential hierarchy. Moreover, characterizations of NC_1 by means of *branching programs* (Barrington's theorem [Bar89]) and PSPACE by means of *bottleneck Turing machines* [CF91] should lead to characterizations of those classes. It is moreover important to characterize not only classes of predicates but classes of functions as well. A natural lead for this is to understand the classes of functions lying in the type $\mathbf{Nat_2} \Rightarrow \mathbf{Nat_2}$ – functions from (binary) natural numbers to (binary) natural numbers – in the models considered. More ge*f*nerally, we expect characterizations and/or definitions of complexity classes of higher-order functionals as the types of higher-order functionals in the models, e.g. $(\mathbf{Nat_2} \Rightarrow \mathbf{Nat_2}) \Rightarrow (\mathbf{Nat_2} \Rightarrow \mathbf{Nat_2})$ for type 2 functionals. As no established theory of complexity for higher-order functional exists at the time, this particular line of research meets item (iii).

(ii) *Show that* Complexity-through-realizability *techniques extend and improve previous work*.

It would be interesting to understand if the definition of algorithms as Abstract State Machines (ASMs) proposed by Gurevich [Gur95] corresponds to a specific case of our definition of algorithms as graphings. Although no previous work attempted to relate ASMs and GoI, an ASM is intuitively a kind of automata on first-order structures and such objects can be seen as graphings [Sei14e]. This expected result will show that the notion of graphing provides an adequate mathematical definition of the notion of algorithm. The *complexity-through-realizability* techniques proposed should also be shown to generalize both the quantitative denotational semantics [LMMP13] and the quantitative realizability technique developed by Hoffman and dal Lago [dLH11]. Building on previous work obtaining a realizability model capturing the ICC logical system ELL [Sei14d], one should also work on demonstrating that *complexity-through-realizability* techniques generalize and extend ICC techniques, for instance by describing the syntax of logical systems corresponding to the realizability models. This should lead to applications such as typing systems for complexity constrained programs and type inference algorithms for statically determining the complexity bounds of programs.

(iii) *Show that* Complexity-through-realizability *techniques apply to a wide variety of computational paradigms*, for instance process calculi, probabilistic/quantum computation, and cellular automata.

First results in this directions are provided by the work sketched in section 4, as it is shown that probabilistic computation can fit into the proposed framework. Moreover, a generalization towards quantum computation seem natural. The framework offered by graphings is indeed particularly fit for those two computational paradigms as it related to operator algebras techniques, hence related to both measure theory and linear algebra. In this particular case of probabilistic and quantum computation, this will allow for characterizations of other standard probabilistic and quantum complexity classes such as PP, BPP, or BQP. In other cases such as concurrent programs or cellular automata where no established complexity

theory exists, this will provide a viable and well-grounded foundation for such a theory.

It is our belief also that important contributions can be made by working on the second objective, namely use well-established mathematical techniques, tools and invariants from operator algebras and dynamical systems for addressing open problems in complexity.

In particular, ℓ^2-Betti numbers [Gab02] appear to be good candidates for this purpose. Let us recall that the notion of microcosm used to characterize complexity classes in the work described above generates a *measured equivalence relation* which in turn defines a von Neumann algebra together with a distinguished maximal abelian sub-algebra. The notion of measured equivalence relation is used to study measurable group actions, and mathematicians have developed fine invariants to classify them, such as *cost* [Gab00] and ℓ^2-Betti numbers[5] [Gab02]. Discussions with operator algebraists lead to some evidence that if two microcosms $\mathfrak{m}$ and $\mathfrak{n}$ have the same[6] ℓ^2-Betti number, the complexity classes characterized are equal. This is coherent with the first results presented in section 4: a number of well-known separation results such as 2DFA(k) $\neq$2DFA(k+1) [Mon76] hold for the various notion of automata considered above, and one can show that the first ℓ^2-Betti numbers of the corresponding microcosms are not equal. We are therefore conjecturing the following result, which would lead to a new method for the obtention of separation results.

Conjecture 1. *Let* $\mathfrak{m},\mathfrak{n}$ *be two microcosms. Then* $\mathfrak{m}$ *and* $\mathfrak{n}$ *have the same* ℓ^2*-Betti numbers if and only if the types* $\mathbf{Nat_2} \Rightarrow \mathbf{Bool}$ *in the corresponding models characterize the same complexity classes.*

I believe moreover that higher-order homotopies relate to higher-order functionals, and I am expecting a more precise result stating that equality of ℓ^2-Betti numbers $\beta_k^{(2)}$ for all $k \leqslant N$ is equivalent to the equality of the characterized type k functionals complexity classes for all $k \leqslant N$.

[5] A definition which is coherent with ℓ^2-Betti number defined by Atiyah [Ati76], the later generalization to groups by Cheeger and Gromov [CG86] reformulated by Lück [Lüc02], and the generalization to von Neumann algebras by Connes and Shlyakhtenko [CS05].

[6] The ℓ^2-Betti number of a microcosm is defined as the ℓ^2-Betti number of the generated measured equivalence relation.

References

[ABPS14] Clément Aubert, Marc Bagnol, Paolo Pistone, and Thomas Seiller. Logic programming and logarithmic space. In Jacques Garrigue, editor, *Programming Languages and Systems - 12th Asian Symposium, APLAS 2014, Singapore, November 17-19, 2014, Proceedings*, volume 8858 of *Lecture Notes in Computer Science*, pages 39–57. Springer, 2014.

[ABS15] Clément Aubert, Marc Bagnol, and Thomas Seiller. Memorization for unary logic programming: Characterizing ptime. *Submitted*, 2015.

[AR02] Andrea Asperti and Luca Roversi. Intuitionistic light affine logic. *ACM Transactions on Computational Logic (TOCL)*, 3(1):137–175, 2002.

[AS14] Clément Aubert and Thomas Seiller. Characterizing co-nl by a group action. *Mathematical Structures in Computer Science*, available as FirstView, December 2014.

[AS15] Clément Aubert and Thomas Seiller. Logarithmic space and permutations. *accepted for publication in Information and Computation*, 2015.

[Ati76] Michael Atiyah. *Elliptic operators, discrete groups and von Neumann algebras*, volume 32-33 of *Astérisque*, pages 43–72. Société Mathématique de France, 1976.

[Bai11] Patrick Baillot. Elementary linear logic revisited for polynomial time and an exponential time hierarchy. In Hongseok Yang, editor, *APLAS*, volume 7078 of *Lecture Notes in Computer Science*, pages 337–352. Springer, 2011.

[Bar89] David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in {NC1}. *Journal of Computer and System Sciences*, 38(1):150 – 164, 1989.

[BC92] Stephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2, 1992.

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

[CF91] JIN-YI CAI and MERRICK FURST. Pspace survives constant-with bottlenecks. *International Journal of Foundations of Computer Science*, 02(01):67–76, 1991.

[CG86] Jeff Cheeger and Mikhael Gromov. l^2-cohomology and group cohomology. *Topology*, 25(2):189–215, 1986.

[CG96] K. J. Compton and E. Grädel. Logical definability of counting functions. *J. Comput. Syst. Sci.*, 53, 1996.

[Cob65] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the 1964 CLMPS*, 1965.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, 1971.

[CS05] Alain Connes and Dimitri Shlyakhtenko. L 2-homology for von neumann algebras. *Journal für die reine und angewandte Mathematik*, 2005(586):125–168, 2005.

[Dan90] Vincent Danos. *La Logique Linéaire Appliquée à l'Étude de Divers Processus de Normalisation (principalement du λ-calcul)*. PhD thesis, University of Paris VII, June 1990.

[DG08] A. Dawar and E. Grädel. The descriptive complexity of parity games. *LMCS*, 5213, 2008.

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

[dLH11] Ugo dal Lago and Martin Hofmann. Realizability models and implicit complexity. *Theoretical Computer Science*, 412:2029–2047, 2011.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *SIAM-AMS Proc.*, volume 7, 1974.

[FK52] Bent Fuglede and Richard V. Kadison. Determinant theory in finite factors. *Annals of Mathematics*, 56(2), 1952.

[FM77] Jacob Feldman and Calvin C Moore. Ergodic equivalence relations, cohomology, and von neumann algebras. I. *Transactions of the American mathematical society*, 234(2):289–324, 1977.

[Gab00] Damien Gaboriau. Coût des relations d'équivalence et des groupes. *Inventiones Mathematicae*, 139:41–98, 2000.

[Gab02] Damien Gaboriau. Invariants l2 de relations d'équivalence et de groupes. *Publ. Math. Inst. Hautes Études Sci*, 95(93-150):15–28, 2002.

[GAL92] Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. The geometry of optimal lambda reduction. In Ravi Sethi, editor, *POPL*, pages 15–26. ACM Press, 1992.

[GG95] E. Grädel and Y. Gurevich. Tailoring recursion for complexity. *Journal of Symbolic Logic*, 60, 1995.

[Ghi07] Dan R. Ghica. Geometry of synthesis: a structured approach to vlsi design. In Martin Hofmann and Matthias Felleisen, editors, *POPL*, pages 363–375. ACM, 2007.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987.

[Gir89a] Jean-Yves Girard. Geometry of interaction I: Interpretation of system F. In *In Proc. Logic Colloquium 88*, 1989.

[Gir89b] Jean-Yves Girard. Towards a geometry of interaction. In *Proceedings of the AMS Conference on Categories, Logic and Computer Science*, 1989.

[Gir95] Jean-Yves Girard. Geometry of interaction III: Accommodating the additives. In *Advances in Linear Logic*, number 222 in Lecture Notes Series, pages 329–389. Cambridge University Press, 1995.

[Gir06] Jean-Yves Girard. Geometry of interaction IV: the feedback equation. In Stoltenberg-Hansen and Väänänen, editors, *Logic Colloquium '03*, pages 76–117, 2006.

[Gir11] Jean-Yves Girard. Geometry of interaction V: Logic in the hyperfinite factor. *Theoretical Computer Science*, 412:1860–1883, 2011.

[Gir12] Jean-Yves Girard. Normativity in logic. In Peter Dybjer, Sten Lindström, Erik Palmgren, and Göran Sundholm, editors, *Epistemology versus Ontology*, volume 27 of *Logic, Epistemology, and the Unity of Science*, pages 243–263. Springer, 2012.

[GS10] Dan R. Ghica and Alex Smith. Geometry of synthesis II: From games to delay-insensitive circuits. *Electr. Notes Theor. Comput. Sci.*, 265:301–324, 2010.

[GS11] Dan R. Ghica and Alex Smith. Geometry of synthesis III: resource management through type inference. In Thomas Ball and Mooly Sagiv, editors, *POPL*, pages 345–356. ACM, 2011.

[GSS92] Jean-Yves Girard, Andre Scedrov, and Philip J. Scott. Bounded linear logic: a modular approach to polynomial-time computability. *Theor. Comput. Sci.*, 97(1):1–66, April 1992.

[GSS11] Dan R. Ghica, Alex Smith, and Satnam Singh. Geometry of synthesis IV: compiling affine recursion into static hardware. In Manuel M. T. Chakravarty, Zhenjiang Hu, and Olivier Danvy, editors, *ICFP*, pages 221–233. ACM, 2011.

[Gur95] Yuri Gurevich. Specification and validation methods. chapter Evolving Algebras 1993: Lipari Guide, pages 9–36. Oxford University Press, Inc., New York, NY, USA, 1995.

[HS65a] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Transactions of the AMS*, 117, 1965.

[HS65b] Juris Hartmanis and Richard Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117, 1965.

[Imm88] Neil Immerman. Nondeterministic space is closed under complementation. In *Structure in Complexity Theory Conference*, 1988.

[JS74] N. Jones and A. Selman. Turing machines and the spectra of first-order formulas. *Journal of Symbolic Logic*, 39, 1974.

[Kle45] Stephen C. Kleene. On the interpretation of intuitionistic number theory. *Journal of Symbolic Logic*, 10, 1945.

[Kri01] Jean-Louis Krivine. Typed lambda-calculus in classical zermelo-fraenkel set theory. *Arch. Mathematical Logic*, 40, 2001.

[Lag09] Ugo Dal Lago. The geometry of linear higher-order recursion. *ACM Trans. Comput. Logic*, 10(2):8:1–8:38, March 2009.

[Lam90] John Lamping. An algorithm for optimal lambda calculus reduction. In Frances E. Allen, editor, *POPL*, pages 16–30. ACM Press, 1990.

[Lev95] Gilbert Levitt. On the cost of generating an equivalence relation. *Ergodic Theory and Dynamical Systems*, 15:1173–1181, 1995.

[LM93] D. Leivant and J.-Y. Marion. Lambda calculus characterizations of poly-time. *Fundam. Inform.*, 19, 1993.

[LM94] Daniel Leivant and Jean-Yves Marion. Ramified recurrence and computational complexity II: Substitution and poly-space. *Lecture Notes in Computer Science*, 933, 1994.

[LMMP13] Jim Laird, Guy McCusker, Giulio Manzonetto, and Michele Pagani. Weighted relational models of typed lambda-calculi. In *Proceedings of the 28th Annual IEEE Symposium on Logic in Computer Science, LICS 2013, June 25-28, 2013, New Orleans, USA*, 2013.

[Lüc02] Wolfgang Lück. *L^2-Invariants: Theory and Applications to Geometry and K-Theory*, volume 44 of *A Series of Modern Surveys in Mathematics*. 2002.

[Mon76] Buckhard Monien. Transformational methods and their application to complexity problems. *Acta Informatica*, 6:95–108, 1976.

[RR97] A. A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55, 1997.

[Sav70] W. Savitch. Relationship between nondeterministic and deterministic tape classes. *Journal of Computer and Systems Sciences*, 4, 1970.

[Sei12a] Thomas Seiller. Interaction graphs: Multiplicatives. *Annals of Pure and Applied Logic*, 163:1808–1837, December 2012.

[Sei12b] Thomas Seiller. *Logique dans le facteur hyperfini : géometrie de l'interaction et complexité*. PhD thesis, Université Aix-Marseille, 2012.

[Sei14a] Thomas Seiller. A correspondence between maximal abelian subalgebras and linear logic fragments. *Submitted*, 2014.

[Sei14b] Thomas Seiller. Interaction graphs: Additives. *Accepted for publication in Annals of Pure and Applied Logic*, 2014.

[Sei14c] Thomas Seiller. Interaction graphs and complexity. Extended Abstract, 2014.

[Sei14d] Thomas Seiller. Interaction graphs: Exponentials. *Submitted*, 2014.

[Sei14e] Thomas Seiller. Interaction graphs: Graphings. *Submitted*, 2014.

[Sei15] Thomas Seiller. Interaction graphs and complexity I. In preparation, 2015.

[SS08] Allan Sinclair and Roger Smith. *Finite von Neumann algebras and Masas*. Number 351 in London Mathematical Society Lecture Note Series. Cambridge University Press, 2008.

[Sze87] R. Szelepscényi. The method of forcing for nondeterministic automata. *Bulletin of the EATCS*, 33, 1987.