Description of the Thesis

Logic in the Hyperfinite Factor: Geometry of Interaction and Complexity

Thomas Seiller*

1 Context

Syntax and Semantics

From a historical point of view, logic is first and foremost the science of reasoning. It is at the beginning of the 20th century that the subject imposed itself for the formalisation of mathematics — we will talk in this case of *mathematical logic*. Traditionally, mathematical logic is a two-headed entity : on one hand sits the syntax which describes how mathematical formulas and proofs can be written — a (usually) finite description of the world —, and on the other hand lies the semantics which study the models of a theory — the mathematical worlds or structures. Syntax is concerned with provability : the rules describing a syntax usually forbids one to write incorrect proofs, and one studies then the formulas which are provable from a given set of axioms. Semantics, on the contrary, is concerned with refutation : to show that a formula A is not a consequence of a given set of axioms one will build a model of this theory in which the formula A does not hold. These two faces of logic are put in correspondance by the *completeness theorem* of Gödel : a formula A is provable if and only if it is satisfied in all models.

Proof Theory

Proof theory, which studies formalisations of proofs, belongs to the syntactic tradition of logic. Several derivation systems have been developed and the differences between logical systems are translated as differences at the level of provability and rules, as it is the case for classical and intuitionnistic logic. At first sight, and for working mathematicians, the most important rule of a derivation system is the *modus ponens*, generalized as the *cut rule*, which allows the use of lemmas : from a proof of $A \Rightarrow B$ and a proof of A, one can obtain a proof of B. However, one of the most important results in proof theory, Gentzen's *Hauptsatz* [Gen64], states that this rule is redondant : if there exists a proof of B, then there exists a proof of B that does not use the cut rule. If this result is important in itself, it became fundamental after the discovery of the proofs-as-programs correspondence, also known as the *Curry-Howard correspondence*.

*Institut des Hautes Études Scientifiques (IHÉS), Le Bois-Marie, 35 route de Chartres, 91440 Bures-sur-Yvette, France

Curry-Howard Correspondance

At the beginning of the XXth century, Church introduced an axiomatisation of the theory of functions : the λ-calculus [Chu41]. It turns out that all computable functions can be represented in λ-calculus, making it an alternative approach to the theory of computation, usually described in terms of *Turing machines* [Tur37]. In this theory, the dynamics of computation is represented as a single rewriting rule, called β-reduction : if t represents a function and u represents an argument, then the application $t(u)$ *reduces* — or β-reduces — to a term r representing the result of the computation of the function represented by t given the argument represented by u. The Curry-Howard correspondence [Cur34, How80] relates the cut-elimination procedure in logic to the β-reduction in λ-calculus. In other terms, it relates the execution of a program to the elimination of "detours", i.e. uses of the cut rule, in a proof. This lead to the study of computation through deduction systems, and in particular through their dynamics — the cut-elimination procedure.

In intuitionnistic logic, the implication $A \Rightarrow B$ can be understood as a function which given any proof of A produces a proof of B. This intuition is reinforced by the Curry-Howard correspondence since a term representing a map from natural numbers $\mathbf{N}$ to natural numbers will correspond to a proof of the formula Nat $\Rightarrow$ Nat, where Nat is a formula representing the set of natural numbers.

Linear Logic

The Curry-Howard correspondence finds some of its most important applications through a back-and-forth between syntax and semantics : starting from a mathematical model of computation, one discovers some operation and pulls it back into the syntax. For instance, Girard introduced linear logic by pulling back into the syntax a natural decomposition of the implication he discovered in a model of the lambda-calculus [Gir88a]. Similarly, Ehrhard and Regnier [ER03, ER06] introduced differential lambda-calculus and differential linear logic by pulling back a notion of differentiation that existed in models of linear logic [Ehr02].

As we just explained, it is through a study of denotational models of λ-calculus [Gir88a] that Girard discovered that the implication $\Rightarrow$ was naturally decomposed in two independent operations. The first operation, the exponentiation — or perennisation — allows the duplication of a proof of A : if a proof of A can be used only once, a proof of $!A$ can be used any number of times. The second operation is a linear implication, written $\multimap$, which produces from a proof a A a proof of B, *consuming A in the process*. The implication $A \Rightarrow B$ is then obtained as the formula $!A \multimap B$. Girard then introduced a syntax — linear logic — which takes into account this semantical decomposition [Gir87].

Linear logic is sensible to ressource consumption, and therefore well-fitted for the study of programs. Indeed, a proof of Nat $\multimap$ Nat in linear logic corresponds to a function from natural numbers to natural numbers that uses its argument only once, i.e. a linear map, while a proof of !Nat $\multimap$ Nat might use its argument any (finite) number of times. The re-use of ressources being associated to the exponential connective !, one may then modify the rules of this connective to obtain restrictions of linear logic which have a computational interest. For instance, Bounded Linear Logic [GSS92], Soft Linear Logic [Laf04] and Light Linear

Logic [Gir95b] are systems in which the proofs of !Nat $\multimap$ Nat correspond to Polytime functions. Elementary Linear Logic [Gir95b, DJ03] characterizes in the same way the functions computable in elementary time.

Proof Nets

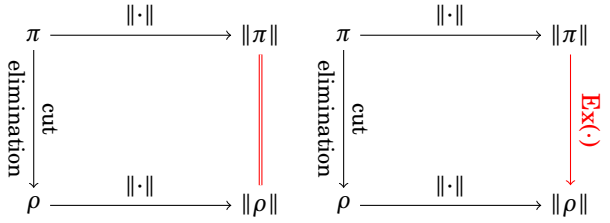
Linear Logic was introduced with two deduction systems — two syntaxes — to represent proofs : a sequent calculus and a kind of natural deduction with multiple conclusions, called proof structures. Proof structures stand appart from traditional deduction systems as the grammar that describes them is too permissive and allows one to write objects that do not correspond to sequent calculus proofs. It is however possible to characterize the proof structures that correspond to proofs — the proof nets — from the others by means of *correctness criterions*. These criterions are usually based on the topology or the geometry of the proof structure in consideration. This particularity of proof structures has been the starting point of Girard's research program known as *geometry of interaction* : a modeling of proofs — or more precisely paraproofs, namely an extended notion of proof — and their dynamics (cut elimination) joint to a test-based reconstruction of logic. Indeed, the extension of syntax introduced by proof structures adds a semantical flavor to the objects in the sense that the added syntactic objects — which are not proofs — somehow play the role of counter-models. As a consequence, one can use these additional objects to *test* the proofs and define an interactive notion of type : a set of paraproofs which is orthogonal to a given set of tests.

Geometry of Interaction : Modelling the Dynamics of Proofs

A Geometry of Interaction (GoI) construction, i.e. a construction that fulfills the GoI research program [Gir89b], is in a first approximation a representation of linear logic proofs that accounts for the dynamics of cut-elimination. A proof is no longer a morphism from A to B — a function from A into B — but an operator acting on the space $A \oplus B$. As a consequence, the modus ponens is longer represented as composition. The operation representing cut-elimination, i.e. the obtention of a cut-free proof of B from a cut-free proof of A and a cut-free proof of $A \multimap B$, consists in constructing the solution to an equation called the *feedback equation* (illustrated in Figure 2). A GoI construction hence represent both the proofs and their normalization. Contrarily to denotational semantics, a proof π and its normalized form π' are not represented by the same object. However, they remain related since the normalization procedure has a semantical counterpart — the execution formula $\mathrm{Ex}(\cdot)$ — which satisfies $\mathrm{Ex}(\pi) = \pi'$. This essential difference between denotational semantics and GoI is illustrated in Figure 1.

Geometry of Interaction : Interactive Reconstruction of Logic

The GoI program is more ambitious than that : the aim is to obtain not only a representation of proofs that accounts for their dynamics, but to reconstruct logical operations from it. As we already mentioned, the objects of study in a GoI construction are a generalization of the notion of proof — paraproofs, in the same



(a) Denotational Semantics (b) Geometry of Interaction

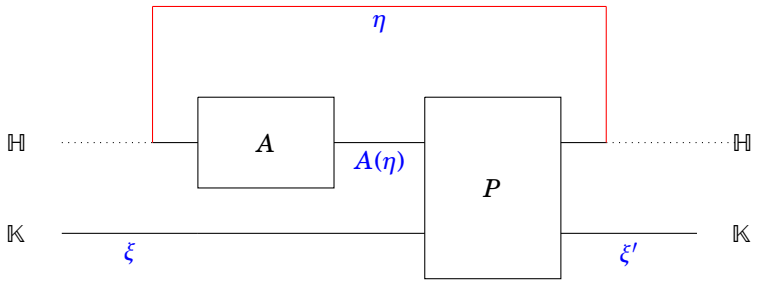
FIGURE 1 – Denotational Semantics vs Geometry of Interaction

$P \in \mathscr{L}(\mathbb{H} \oplus \mathbb{K})$ represents[a] a program/proof of implication
$A \in \mathscr{L}(\mathbb{H})$ represents an argument.
$R \in \mathscr{L}(\mathbb{K})$ represents the result of the computation if :

$$R(\xi) = \xi' \Leftrightarrow \exists \eta, \eta' \in \mathbb{H}, \begin{cases} P(\eta \oplus \xi) & = & \eta' \oplus \xi' \\ A(\eta') & = & \eta \end{cases}$$

a. Here, $\mathbb{H}$ and $\mathbb{K}$ are separable infinite-dimensional Hilbert spaces, and $\mathscr{L}(\star)$ denotes the set of operators acting on the Hilbert space $\star$: bounded (or, equivalently, continuous) linear maps from $\star$ to $\star$.

(a) Formal statement



(b) Illustration of the equation

FIGURE 2 – The Feedback Equation

sense the proof structure where a generalization of the notion of sequent calculus proof. This point of view allows a reconstruction of logic as a description of how paraproofs interact. It is therefore a sort of "discursive syntax" where paraproofs are opposed one to another, where they argue together in a way reminiscent of game semantics, each one trying to prove the other wrong. This argument terminates when one of them gives up. The discussion itself corresponds to the execution formula, which describes the solution to the feedback equation and generalizes the cut-elimination procedure to this generalized notion of proofs. Two paraproofs are then said *orthogonal* — denoted by the symbol $\perp$ — when this arguement (takes place and) terminates. A notion of formula is then drawn from this notion of orthogonality : a formula is a set of paraproofs A equal to its bi-orthogonal closure $A^{\perp\perp}$ or, equivalently, a set of paraproofs $A = B^{\perp}$ which is the orthogonal to a given set of paraproofs B.

Drawing some intuitions from the Curry-Howard correspondence, one may propose an alternative reading to this construction in terms of programs. Since proofs correspond to well-behaved programs, paraproofs are a generalization of those, representing somehow *badly-behaved programs*. If the orthogonality relation represents negation from a logical point of view, it represents a notion of *testing* from a computer science point of view. The notion of formula defined from it corresponds to a notion of type, defined interactively from how (para)programs behave. This point of view is still natural when thinking about programs : a program is of type nat → nat because it produces a natural number when given a natural number as an argument. On the logical side, this change may be more radical : a proof is a proof of the formula Nat ⇒ Nat because it produces a proof of Nat each time it is cut (applied) to a proof of Nat.

Geometry of Interaction : Logic issued from Proofs/Programs

Once the notion of type/formula defined, one can reconstruct the connectives : from a "low-level" — between paraproofs — definition, one obtains a "high-level" definition — between types. For instance, the connective $\otimes$ is first defined between any two paraproofs $\mathfrak{a}, \mathfrak{b}$, and this definition is then extended to types by defining $A \otimes B = \{\mathfrak{a} \otimes \mathfrak{b} \mid \mathfrak{a} \in A, \mathfrak{b} \in B\}^{\perp\perp}$. As a consequence, the connectives are not defined in an *ad hoc* way, but their definition is a consequence of their computational meaning : the connectives are defined on proofs/programs and their definition at the level of types is just the reflection of the interaction between the execution — the dynamics of proofs — and the low-level definition on paraproofs. Logic thus arises as generated by computation, by the normalization of proofs : types/formulas are not there to tame the programs/proofs but only to describe their behavior. This is reminiscent of realizability in the sense that a type is defined as the set of its (para-)proofs. Of course, the fact that we consider a generalized notion of proofs from the beginning has an effect on the construction : contrarily to usual realizability models (except from classical realizability in the sense of Krivine), the types A and $A^{\perp}$ (the negation of A) are in general both non-empty. This is balanced by the fact that one can define a notion of successful paraproofs, which corresponds in a way to the notion of winning strategy in game semantics, This notion on paraproofs then yields a high-level definition : a formula/type is *true* when it contains a successful paraproof.

Geometry of Interaction : Between Syntax and Semantics

A GoI construction is therefore neither a syntax, as it is too expressive and gain from this a semantical flavor, neither a semantics, as it is too restrictive and therefore keep some of the good properties of syntax. A GoI construction provides an alternative, more homogeneous, approach to the traditional proofs *vs.* models — syntax *vs.* semantics — opposition. It is a refoundation of logic in its whole, where the usual theorems and properties of syntax, semantics, as well as properties relating both them, can be stated. As an example of a purely syntactic property, the Church-Rosser property is there understood as the associativity of the execution formula, or rather conversely, the associativity of the execution formula in GoI plays the role of the Church-Rosser property [1] :

$$\mathrm{Ex}(\mathrm{Ex}(F,A),B) = \mathrm{Ex}(\mathrm{Ex}(F,B),A)$$

In a similar way, the notion of completeness finds its counterpart in GoI as the property called the *internal completeness* of a connective. For instance, the connective $\otimes$ is internally complete if $A \otimes B$ (as defined above) is equal to the set $\{\mathfrak{a} \otimes \mathfrak{b} \mid \mathfrak{a} \in A, \mathfrak{b} \in B\}$, i.e. if every paraproof of $A \otimes B$ (every proof of $A \otimes B$, every model satisfying $A \otimes B$) is obtained as the tensor of a paraproof of A and a paraproof of B (is obtained from a tensor rule between a proof of A and a proof of B, satisfies A and satisfies B).

Geometry of Interaction in the Hyperfinite Factor

Since the introduction of the GoI research program [Gir89b], a number of constructions have been proposed by Girard in order to fulfill it [Gir89a, Gir88b, Gir95a, Gir11]. In each of these construction, the notion of paraproof remains the same [2] : an operator acting on a (separable) Hilbert space. In the first models, the execution was represented as the so-called *execution formula*, which was based upon the inversion of an operator. More recently, Girard showed that while the execution was not always defined, the solution to the feedback equation still exists and is unique (and satisfies associativity) as long as the operators involved are of norm at most 1 [Gir06]. This can be stated as the following theorem.

Theorem 1 (Girard [Gir06])**.** *If $a \in \mathscr{L}(\mathbb{H} \oplus \mathbb{K})$, $b \in \mathscr{L}(\mathbb{H})$ are operators of norm at most* 1*, the solution to the feedback equation involving a and b exists, is unique, and is an operator of norm at most* 1 *in the von Neumann algebra generated by a and b.*

This difficult and very technical result lead him to a new construction of a geometry of interaction [Gir11] in which the operators considered are elements of a chosen von Neumann algebra [3] : the hyperfinite factor of type II_1. The notion of orthogonality is then defined using a generalization of the determinant of matrices called the Fuglede-Kadison determinant [FK52]. This construction will be referred to as *hyperfinite GoI* in the following.

1. We use here the notation $\mathrm{Ex}(A,B)$ (two arguments) while we used $\mathrm{Ex}(\pi)$ (one argument) beforehand. This slight difference comes from the fact that a proof is interpreted as a couple, a technical detail we did not want to mention in order to simplify the discussion.

2. Even though the paper GoI3 is not using operators, it can be naturally presented in this way.

3. For technical reasons, they are chosen in this algebra but embedded in a bigger algebra : the hyperfinite factor of type II_∞.

First GoI Constructions and Computer Science

Since it offers a mathematical model of the execution of programs, the GoI program naturally arose as a well-suited tool for the study of computational complexity. For instance, it is by using the first construction of a geometry of interaction [Gir89a] that Abadi, Gonthier and Lévy [GAL92] showed the optimality of Lamping's reduction in lambda-calculus [Lam90]. The same construction was also applied in implicit computational complexity [BP01], and was the main inspiration behind dal Lago's context semantics [Lag09].

In spite of their seemingly deep abstraction, the GoI constructions offer a mathematical model which is very close to actual computing. As an illustration of this fact, let us mention the *Geometry of Synthesis* program initiated by Ghica [Ghi07, GS10, GS11, GSS11]. This research program, inspired by geometry of interaction, aims at obtaining logical synthesis methods for VLSI (Very Large Systems Integration) designs.

2 Results presented in the thesis

2.1 Overview

This thesis is a study of the GoI constructions, with an emphasis on the hyperfinite GoI construction and its applications to computational complexity. I develop a combinatorial framework — called interaction graphs — which simultaneously generalizes, unifies and simplifies all the previously known GoI constructions. I then show that the hyperfinite GoI construction, as well as the combinatorial approach just mentioned, provide a framework in which one can study both time and space complexity.

Interaction Graphs

The Interaction Graphs provide a combinatorial framework that fulfills the GoI research program. Proofs — or rather paraproofs — are represented by graphs, while the execution/normalization corresponds to the construction of the graph of alternating paths between two given graphs (illustrated by Figure 3). I showed that this operation constructs the (necessarily unique) solution to the feedback equation (Figure 2). The whole construction is parametrized by a map, and therefore abstracts a whole family of models. This framework :

— unifies the previously introduced GoI constructions : those are recovered from insightful choices of the parameter ;

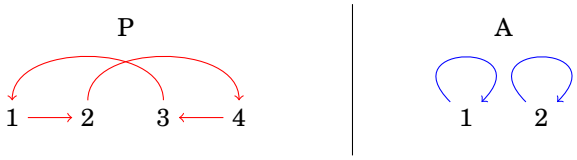
— generalizes these constructions : the framework is very flexible and can be extended in many ways, it moreover avoids the constantly constraining issue of divergence that one has to deal with when working with operators ;

— simplifies these constructions : the operators are replaced by graphs [4], and the interpretations of proofs are finitely representable.

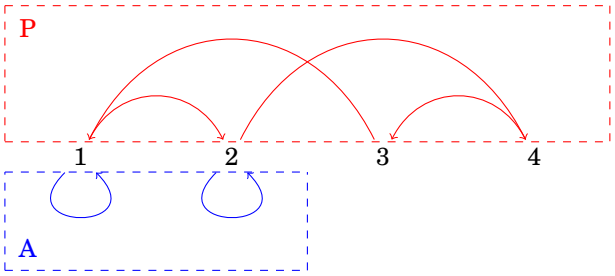
Moreover, this framework brings new insights about the GoI program. In particular :

— from a theoretical point of view, it puts into light a relation between cycles and paths upon which all previously introduced GoI constructions are

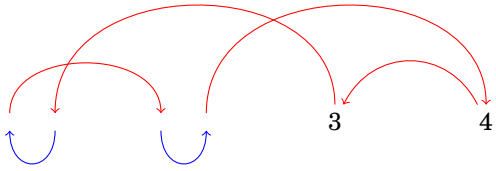
4. When dealing with exponential connectives, we consider a generalization of graphs named *graphings* which is in some way a *geometric realization* of a graph.



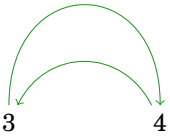
(a) Graphs P and A



(b) Graph $P \square A$ representing the application of A to P



(c) Alternating paths in $P \square A$



(d) Graph R representing the result of the execution of P applied to A

FIGURE 3 – Modeling execution (Interaction Graphs)

founded. As a consequence, the *adjunctions* [5] which insure the correctness of Girard's constructions are obtained as a corollary of this previously unknown combinatorial identity ;

— from a practical point of view, we obtained a more down-to-earth model based on a tractable notion of objects. This allows to obtain better insights on how the constructions can be used in complexity theory.

Applications to Complexity

The second part of the thesis is concerned with the applications of the GoI program to computational complexity. I show in this part that both time and space complexity can be dealt with, focusing on the hyperfinite GoI construction and the Interaction Graphs framework. Theses results can be thought as part of *implicit computational complexity* (ICC), a field which studies languages and computational devices constrained in their use of their ressources which corres-

5. This terminology, used by Girard, may lead a reader knowing about category-theory to a confusion. This notion of adjunction is *not* the categorical notion of adjunction but refers to a specific property in GoI constructions.

pond to complexity classes such as PSpace, Ptime, LogSpace, etc. The aim of ICC is to study algorithmic complexity without making reference to external measure conditions or particular models of computation, but only in terms of restrictions of languages and computational principles. There are many ways in which such constraints can be expressed, among which the constrained linear logic systems such as Bounded Linear Logic, Light Linear Logic, etc.

The first result I obtained is about time complexity classes and is related to known results and works in ICC. I showed a *soundness* result for Elementary Linear Logic in both the frameworks of the hyperfinite GoI of Girard and the interaction graphs. This result, which mainly states that one can interpret *correctly* proofs of Elementary Linear Logic in these frameworks, shows that the latter are well-suited for the study of time complexity. Indeed, a result of Baillot [Bai11] shows that it is possible to characterize the exponential hierarchy (including Ptime) as types in ELL.

On the other hand I developed, in a joint work with C. Aubert, a completely new approach to computational complexity that was proposed by Girard [Gir12]. Based on insights and ideas from the hyperfinite GoI construction, this approach is based on the use of the crossed product [6] construction to characterize complexity classes as sets of operators on the hyperfinite factor of type II_1. Introducing a suited notion of abstract machines — the pointer machines, we show how they can be represented by operators induced by a group action. We use here the setting of Interaction Graphs in order to explain how the construction relates to the GoI program. This work corresponds to the last chapter of the thesis : we show how this approach yields a characterization of the non-deterministic space complexity class NL.

2.2 Interaction Graphs

The Trefoil Property [Sei12c, Ch. 5] [Sei12a]

Departing from the realm of infinite-dimensional vector spaces and linear maps between them, we propose a graph-theoretical construction where (para-)proofs are interpreted by finite objects [7]. The graphs we consider are directed and weighted, where the weights are taken in a monoid $(\Omega, \cdot)$.

Definition 1. A *directed weighted graph* is a tuple G, where V^G is the set of vertices, E^G is the set of edges, s^G and t^G are two functions from E^G to V^G, the *source* and *target* functions, and ω^G is a function $E^G \to \Omega$.

The construction is centered around the notion of alternating paths. Given two graphs F and G, an alternating path is a path $e_1 \dots e_n$ such that $e_i \in E^F$ if and only if $e_{i+1} \in E^G$. The set of alternating paths will be used to define the interpretation of cut-elimination in the framework, i.e. the graph $F::G$ — the *execution of F and G* — is defined as the graph of alternating paths between F and G whose source and target are in the symmetric difference $V^F \Delta V^G$. The weight of a path is naturally defined as the product of the weights of the edges it contains.

6. Or rather a particular case of it : the wreath product construction.

7. Even though the graphs we consider can have an infinite set of edges, linear logic proofs are represented by finite graphs (disjoint unions of transpositions).

As it is usual in mathematics, this notion of paths cannot be considered without the associated notion of cycle : an *alternating cycle* between two graphs F and G is a cycle which is an alternating path $e_1 e_2 \dots e_n$ such that $e_1 \in V^F$ if and only if $e_n \in V^G$. For technical reasons, we actually consider the related notion of 1-circuit, which is a cycle satisfying some technical property. We denote by $\mathscr{C}(F,G)$ the set of 1-circuits in the following. We show that these notions of paths and cycles satisfy a property we call the *trefoil property* which will turn out to be fundamental. We also exhibit several other notions of graphs with their associated notions of paths and cycles, proving in each case that the trefoil property holds.

We will use Figure 4 to explain the trefoil property. In this figure, we consider three graphs F, G, and H such that a given vertex (in any of the graphs) cannot be a vertex in the three graphs simultaneously, i.e. the intersection $V^F \cap V^G \cap V^H$ is empty. The double arrows in Figure 4(a) represent the sets of edges (in any of the graphs) that one can go through to go from one graph to the other : for instance the double arrow between V^F and V^G stands for the set of edges of F whose target is an element of V^G. We also represent the sets of cycles formed from edges of F and G only (respectively edges of F and H only, respectively edges of G and H only) by a red cycle (respectively blue, respectively green) in Figures 4(b), 4(c), 4(d), and 4(e). Finally, the set of cycles that contain at least one edge from each graph is represented by a violet cycle in these figures. The coloured rectangles in Figures 4(c), 4(d), and 4(e) represent the result of the execution between two graphs, which one can understand as a box whose content is unknown. We then notice that during the execution of two graphs, say F ans G, one "loses" the alternating cycles composed of edges of F and G only. The trefoil property then states that :

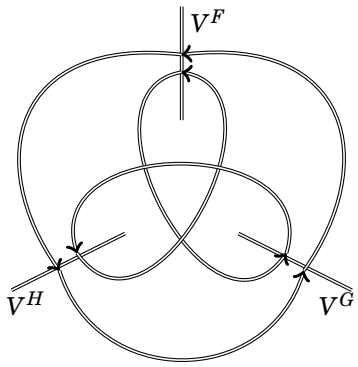
$$\mathscr{C}(F::G,H) \cup \mathscr{C}(F,G) \cong \mathscr{C}(G::H,F) \cup \mathscr{C}(G,H) \cong \mathscr{C}(H::F,G) \cup \mathscr{C}(H,F)$$

where $\cong$ denotes a weight-preserving bijection between the sets of 1-circuits.

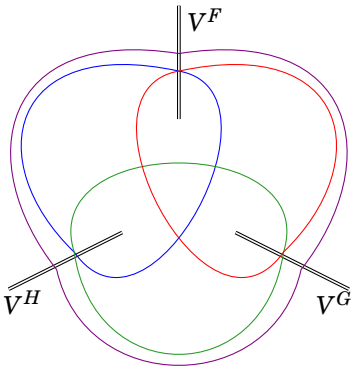
Multiplicative-Additive Linear Logic [Sei12c, Ch. 6-7] [Sei12b] [Sei12a]

We then show how one can define the multiplicative and additive connectives of Linear Logic, obtaining a construction that fulfills the GoI research program. This construction is moreover parametrized by a map from the set Ω to $\mathbb{R}_{\geqslant 0} \cup \{\infty\}$, and therefore yields not only one but a whole family of models. This parameter is introduced to define the notion of orthogonality in our setting. Indeed, given a map m and two graphs F,G we define $[\![F,G]\!]_m$ as the sum $\sum_{\pi \in \mathscr{C}(F,G)} m(\omega(\pi))$, where $\omega(\pi)$ is the weight of the cycle π.

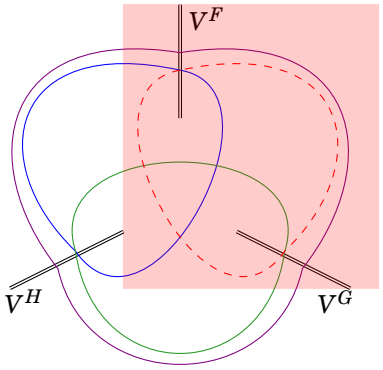
We moreover show how, from any of these constructions, one can obtain a $*$-autonomous category $\mathfrak{Graph}_{MLL}$ with $\parr \ncong \otimes$ and $1 \ncong \bot$, i.e. a non-degenerate denotational semantics for Multiplicative Linear Logic (MLL). However, as in all the versions of GoI dealing with additive connectives, our construction of additives does not define a categorical product. We solve this issue by introducing a notion of *observational equivalence* within the model. We are then able to define a categorical product from our additive connectives when considering classes of observationally equivalent objects, thus obtaining a denotational semantics for Multiplicative Additive Linear Logic (MALL).



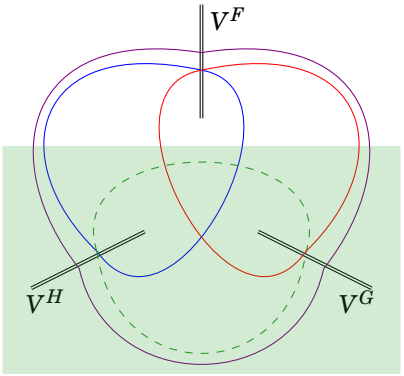
(a) Representation of F, G, H



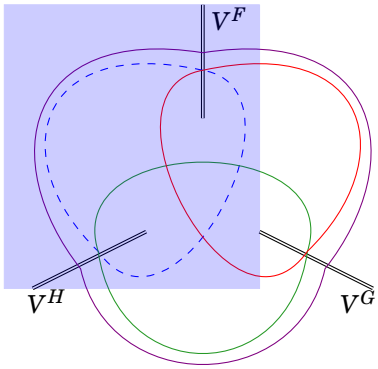
(b) Alternating cycles between F, G and H



(c) Alternating cycles between $F::G$ and H



(d) Alternating cycles between $G::H$ and F



(e) Alternating cycles between $F::G$ and H

FIGURE 4 – Graphical representation of the trefoil property

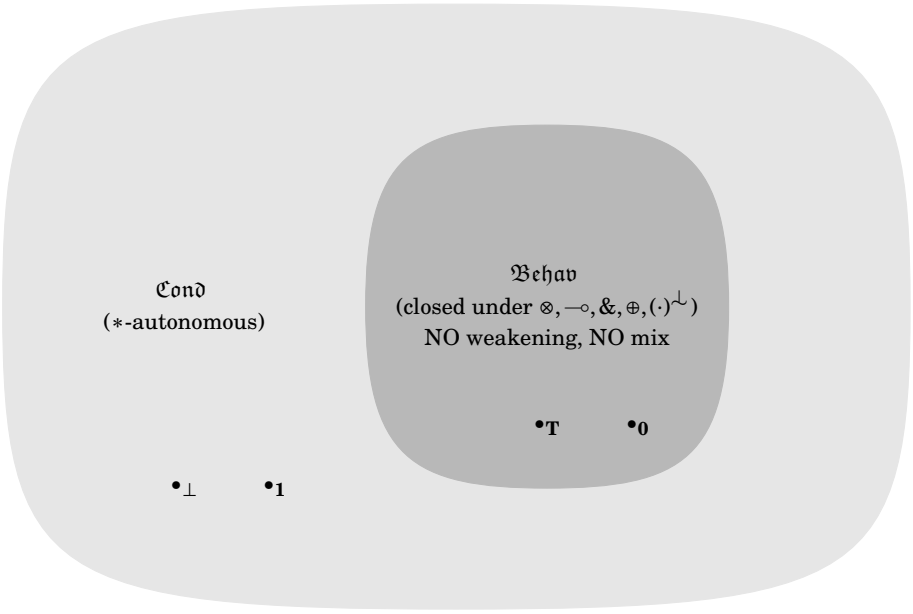


FIGURE 5 – The categorical models

One important point in this work is the fact that all results rely on a single geometric property, the previously introduced *trefoil property*, which describes how the sets of 1-circuits evolve during an execution. This property insures on its own the four following facts :

— we obtain a $*$-autonomous category $\mathfrak{Graph}_{MLL}$;
— the observational equivalence is a congruence on this category ;
— the quotiented category $\mathfrak{Cond}$ inherits the $*$-autonomous structure ;
— the quotiented category $\mathfrak{Cond}$ has a full subcategory $\mathfrak{Behav}$ with products.

This can be summarized in the following two theorems.

Theorem 2. *For any map $m : \Omega \to \mathbf{R} \cup \{\infty\}$, the categories $\mathfrak{Cond}$ and $\mathfrak{Graph}_{MLL}$ are non-degenerate categorical models of Multiplicative Linear Logic with multiplicative units.*

Theorem 3. *For any map $m : \Omega \to \mathbf{R} \cup \{\infty\}$, the full subcategory $\mathfrak{Behav}$ of $\mathfrak{Cond}$ is a non-degenerate categorical model of Multiplicative-Additive Linear Logic with additive units.*

The categorical model we obtain has two layers (see Figure 5). The first layer consists in a non-degenerate (i.e. $\otimes \neq \parr$ and $\mathbf{1} \neq \perp$) $*$-autonomous category $\mathfrak{Cond}$, hence a denotational model for MLL with units. The second layer is the full subcategory $\mathfrak{Behav}$ which does not contain the multiplicative units but is a non-degenerate model (i.e. $\otimes \neq \parr$, $\oplus \neq \&$ and $\mathbf{0} \neq \top$) of MALL with additive units that does not satisfy the mix and weakening rules.

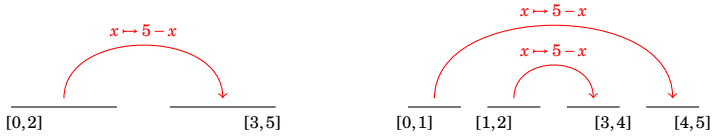


FIGURE 6 – A graphing and one of its refinements

Graphings and Exponentials [Sei12c, Ch. 8] [Sei13]

In the following chapter, a generalization of graphs is introduced. This generalization allows one to define exponentials and second order quantification. The main point about this generalization is that a vertex can always be cut in an arbitrary (finite) number of sub-vertices, with the idea that these sub-vertices are smaller (hence vertices have a *size*) and form a partition of the initial vertex (where two sub-vertices have the same size). These notions could be introduced and dealt with combinatorially, but we chose to use measure-theoretic notions in order to ease the intuitions and some proofs. In fact, as we show at the beginning of the chapter, a *graphing* — the notion which is introduced as a generalization of the notion of graph — can be though of and used as a graph.

Definition 2 (Weighted Graphing)**.** Let $X = (X, \mathscr{B}, \lambda)$ be a measured space and $V^F \in \mathscr{B}$ a set of finite measure. A *graphing over* X 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 $e \in E^F$, ω_e^F is an element of Ω, and ϕ_e^F is a measure-preserving transformation between the measurable sets $S_e^F \subset V^F$ and $T_e^F \subset V^F$.

It is natural, as we are working with measure-theoretic notions, to identify two graphings that differ only on a set of null measure. This leads to the definition of an equivalence relation between graphings : that of *almost everywhere equality*. Moreover, since we want vertices to be *decomposable* into any finite number of parts, we want to identify a graphing G with the graphing G' obtained by replacing an edge $e \in E^F$ by a finite family of edges $e_i \in G'$ $(i = 1, \ldots, n)$ subject to the conditions :

— the family $\{S_{e_i}^{G'}\}_{i=1}^n$ (resp. $\{T_{e_i}^{G'}\}_{i=1}^n$) is a partition of S_e^G (resp. T_e^G) ;

— for all $i = 1, \ldots, n$, $\phi_{e_i}^{G'}$ is the restriction of ϕ_e^G on $S_{e_i}^{G'}$.

Such a graphing G' is an example of a *refinement of* G, and one can easily generalize the previous conditions to define a general notion of refinement of graphings. Figure 6 gives the most simple example of refinement. To be a bit more precise, we define, in order to ease the proofs, a notion of refinement *up to almost everywhere equality*. We then define a second equivalence relation on graphings by saying that two graphings are equivalent if and only if they have a common refinement (up to almost everywhere equality).

The objects under study are thus equivalence classes of graphings modulo this equivalence relation. Most of the technical results on graphings contained in this chapter aim at showing that these objects can actually be manipulated as graphs : one can define paths and cycles and these notions are coherent with the quotient by the equivalence relation just mentioned. As a consequence, given two graphings F, G one can define their execution $F :: G$ and a measurement of their interaction $[\![F, G]\!]_m$, where m is a "measure" of 1-circuits. We then pinpoint

the notion of *circuit-measuring map* and show that for any choice of such maps the execution and interaction satisfy the trefoil property. This implies, from the previous chapters, that from this framework arises another family of models of multiplicative-additive linear logic.

These models are however more interesting since one can define second-order quantifiers and exponential connectives. As an example, we explain how one can define an exponential connective ! which satisfies the *functorial promotion rule*, i.e. the main rule governing this connective in linear logic. The end of the chapter explains how one can interpret *elementary linear logic*, i.e. a restrained linear logic which captures the set of functions computable in elementary time, by means of graphings. This leads to the following theorem, which is true for any interpretation of elementary linear logic proofs and formulas.

Theorem 4. *For any proof π of a sequent $\vdash \Gamma$ in Elementary Linear Logic, the interpretation $\|\pi\|$ of π is a successful element in the interpretation $\|\vdash \Gamma\|$ of the sequent $\vdash \Gamma$.*

Relations to Girard's Constructions [Sei12c, Ch. 9] [Sei12b] [Sei12a]

We then study two particular constructions. Namely, we restrict ourselves to graphs with weights in $]0,1]$ endowed with the usual multiplication and to the parameters $m(x) = \infty$ and $m(x) = -\log(1-x)$. The underlying intuition is that these two particular models are combinatorial versions of Girard's GoI constructions. To make this correspondence formal, we first define a matrix associated to any graph.

Definition 3. Let G be a weighted graph. The matrix $\mathscr{M}_G$ is the weight matrix of the contracted graph $\hat{G}$ defined as [8] :

$$\begin{aligned} V^{\hat{G}} &= V^G \\ E^{\hat{G}} &= \{(v,w) \mid E^G(v,w) \neq \emptyset\} \\ \omega^{\hat{G}} &= (v,w) \mapsto \sum_{e \in E^G(v,w)} \omega(e) \end{aligned}$$

In some cases, the obtained object is not a matrix, since some of the sums used to compute weights may diverge. We thus stick to those graphs G for which the obtained $\mathscr{M}_G$ is a matrix of norm at most 1. This is coherent with the similar restriction that appears in Girard's constructions. We then relate the model obtained for $m(x) = -\log(1-x)$ to Girard's hyperfinite GoI where A, B were defined to be orthogonal if and only if $-\log(\det^{FK}(1-AB)) \neq 0, \infty$, where $\det^{FK}$ denotes the Fuglede-Kadison determinant [FK52].

Theorem 5. *Let F, G be weighted graphs and m be the map $x \mapsto -\log(1-x)$. Then :*

$$[\![F,G]\!]_m = \sum_{\pi \in \mathscr{C}(F,G)} -\log(1-\omega(\pi)) = -\log(\det(1-\mathscr{M}_F\mathscr{M}_G))$$

We then define a map Ψ which associates to each "graphs paraproof" an "hyperfinite paraproof". As a consequence of the last theorem and some other technical results, we are able to show that the map Ψ from the interaction graphs to

8. We write $E^G(v,w)$ the set of edges in G whose source is v and whose target is w. We moreover omit the definition of the source and target maps which are straightforward.

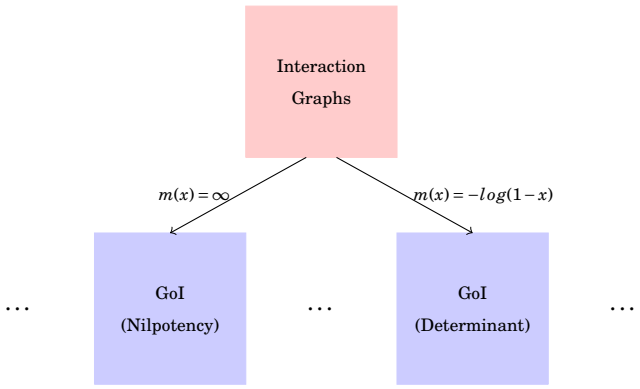


FIGURE 7 – Interaction Graphs and Girard's Constructions

the hyperfinite GoI is a "morphism", i.e. it is coherent with the GoI constructions of the connective $\otimes, \&$, with the execution :: and the orthogonality relation $\perp\!\!\!\perp$.

Theorem 6. *Let* $m(x) = -\log(1-x)$*. Then, if* $\mathfrak{a}, \mathfrak{b}$ *denote paraproof in the interaction graph setting :*

$$\begin{aligned} \Psi(\mathfrak{a} :: \mathfrak{b}) &= \Psi(\mathfrak{a}) :: \Psi(\mathfrak{b}) \\ \Psi(\mathfrak{a} \otimes \mathfrak{b}) &= \Psi(\mathfrak{a}) \otimes \Psi(\mathfrak{b}) \\ \Psi(\mathfrak{a} \,\&\, \mathfrak{b}) &= \Psi(\mathfrak{a}) \,\&\, \Psi(\mathfrak{b}) \\ \mathfrak{a} \perp\!\!\!\perp \mathfrak{b} &\Leftrightarrow \Psi(\mathfrak{a}) \perp\!\!\!\perp \Psi(\mathfrak{b}) \end{aligned}$$

where the right-hand operations are the hyperfinite GoI constructions.

The second model considered, where one uses the "dull" map $m(x) = \infty$, is shown to correspond in the same way to the older constructions of Girard [Gir89a, Gir88b, Gir95a], where orthogonality $A \perp\!\!\!\perp B$ was defined in terms of the nilpotency of AB. Figure 7 illustrates the results obtained in this chapter : among the numerous constructions obtained, we recover Girard's constructions for adequate choices of the parameter map.

2.3 Computational Complexity

Time Complexity [Sei12c, Ch. 10]

The next chapter is then centered around the notion of *subjective truth*, and studies the interpretations of sequent calculus proofs of linear logic in the hyperfinite GoI framework. This notion was introduced by Girard because the usual notion of truth, that one can define in older GoI constructions, was in some sense basis-dependent (here *basis* denotes a Hilbert basis). Since the hyperfinite GoI is not basis-dependent, one has to define a notion of *viewpoint* which generalizes the notion of basis to the hyperfinite type II_∞ factor. Then, the notion of success, and therefore of truth, depends on the chosen viewpoint.

We first show that the notion of viewpoint is related to the notion of *maximal abelian subalgebra* (MASA), a well-studied field in the domain of operator

algebras. In doing so, we exhibit a connection between invariants of these subalgebras and the expressivity of the fragments of linear logic one can interpret. In particular, we use an invariant described by Dixmier [Dix54] who considered three types of MASAs (regular, semi-regular and singular) according to the algebra generated by their normalizer. The results obtained can be summarized in the following theorem.

Theorem 7. *Let $\mathscr{A}$ be a maximal abelian subalgebra. If $\mathscr{A}$ is :*
- *singular then any interpretation of proofs w.r.t. $\mathscr{A}$ is trivial ;*
- *semi-regular then there is a (non-trivial) sound interpretation of MALL w.r.t. $\mathscr{A}$;*
- *regular then there is a (non-trivial) sound interpretation of ELL w.r.t. $\mathscr{A}$;*

The results shown in the thesis are actually a bit stronger in the case of regular von Neumann subalgebras. As a consequence of a result from Connes, Feldmann and Weiss [CFW81], we show that any exponential connective can be soundly interpreted w.r.t $\mathscr{A}$ in this case, even though we restrict our study to the exponential defined by Girard. This chapter also contains discussions about a finer invariant of MASAs, called the Pukansky invariant [Puk60], which is related to the interpretation of exponential connectives.

To sum up, we have shown that Elementary Linear Logic can be correctly interpreted in this setting, obtaining by the way a fine analysis of the different interpretations that are possible according to the different choices of viewpoints.

Space Complexity [Sei12c, Ch. 11] [AS12]

The last chapter contains results obtained from a collaboration with C. Aubert, at the time a PhD candidate in Paris 13 University. This chapter is concerned with a new approach to computational complexity proposed by Girard [Gir12] and inspired from the hyperfinite GoI construction. We begin by a careful study of the representation of binary lists in the interaction graphs setting. These representations yields operators in the hyperfinite type II_1 factor $\mathfrak{R}$ through the "morphism" exhibited earlier in the thesis. We then consider the set of such representations as the set of "input tapes" for an abstract machine model yet to be defined.

To obtain an adequate notion of machine, one has to be careful in the definitions, since a single binary list has many different representations, and a suitable notion of machine should not distinguish two distinct representations. Following the idea proposed by Girard, we show that one can obtain such a suitable notion by using the wreath product construction of operator algebras, a particular case of the *crossed product construction*. This wreath product, denoted $(\bigotimes_{g\in G}\mathfrak{R})\rtimes\mathscr{G}$, is constructed from an algebra $\mathfrak{R}$ and a group G. It contains two sub-algebras that will be of particular interest to us :
- $\mathfrak{N}$, the first copy of $\mathfrak{R}$ in the infinite tensor product ;
- $\mathfrak{G}$ the algebra generated by the representations — the internalizations — of the action of $\mathscr{G}$ onto the infinite tensor product defined as

$$h \mapsto (\bigotimes_{g\in G} x_g \mapsto \bigotimes_{g\in G} x_{h^{-1}g})$$

The following theorem then states that the operators in the algebra $\mathfrak{G}$ act uniformly on the set of representations of a given integer, and therefore provides a suitable notion of machines.

Theorem 8. *Let $N, M \in \mathfrak{N}$ be two representations of an integer, and ϕ an element of $\mathfrak{G}$. Then $N\phi$ is nilpotent if and only if $M\phi$ is nilpotent.*

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

Definition 4. Let $\phi \in \mathfrak{M}_6(\mathbf{C}) \otimes \mathfrak{G} \otimes \mathfrak{M}_k(\mathbf{C})$ be an observation. We define the language accepted by ϕ by (N_n denotes any representation of the integer n) :

$$[\phi] = \{n \in \mathbf{N} \mid \phi(N_n \otimes 1) \text{ nilpotent}\}$$

By extension, if P is a set of observations, we denote by $[P]$ the set $\{[\phi] \mid \phi \in P\}$.

To sum up the construction, one has the following objects :
— an algebra containing representations of integers : the hyperfinite factor $\mathfrak{R}$ of type II_1, embedded in $\mathfrak{K}$ through the morphism $\pi \circ \iota_0$;
— an algebra containing the representations of machines, or algorithms : the von Neumann sub-algebra $\mathfrak{G}$ of $\mathfrak{K}$ generated by the set of unitaries $\{\lambda(\sigma) \mid \sigma \in \mathscr{S}\}$;
— a notion of acceptance : if N is a representation of a integer and ϕ is a representation of a machine, we say that ϕ accepts N when ϕN is nilpotent.

We then exhibit two sets of operators P_+ and $P_{\geqslant 0}$ in the algebra $\mathfrak{G}$, and introduce a notion of abstract machine we call *pointer machines* [9]. We show how these machines can be represented faithfully as operators in P_+ and $P_{\geqslant 0}$, which gives an inclusion of the class co-NL into the classes $[P_+]$ and $[P_{\geqslant 0}]$.

The next part consists in the proof of a technical lemma that states that for every integer representation N and every operator in $P_{\geqslant 0}$ (which contains P_+), there exists two matrices (operators acting on a finite-dimensional Hilbert space) M and $\bar{\phi}$ such that $N\phi$ is nilpotent if and only if $M\bar{\phi}$ is nilpotent. This shows that one can check the nilpotency of matrices in order to decide wether the operator ϕ accepts the integer represented by N. An easy argument then shows that checking the nilpotency of the product $M\bar{\phi}$ can be done with a non-deterministic Turing machine using logarithmic space. This converse inclusion therefore yields the main theorem [10]

Theorem 9.

$$\text{co-NL} = [P_+] = [P_{\geqslant 0}]$$

3 Later Results

The work in my thesis have led to later results, in particular concerning complexity theory. The results presented in the last chapter were extended to obtain a characterization of the class L, exhibiting a relation between determinism and an norm on operators [AS13]. This work also showed that our notion of pointer machines and the more usual notion of two-way multi-head automata [11] [Har72] are equivalent models of computation.

9. This notion of machines was later shown to be equivalent to the *two-way multi-head automata [AS13].*

10. Although it is known that the classes co-NL and NL are equal [Imm88], we chose not to use this result in our work. This leaves open the possibility of obtaining a new proof of this result in our setting. It could also lead to a result in operator algebras mirroring this theorem.

11. This variant of automata is known to provide a characterization of NL and L [HKM08].

More recently I showed, using the interaction graphs framework, that the notion of orthogonality in the hyperfinite GoI and the notion of interaction used to characterize coNL and L are the same. The contributions of this work are :

— the obtention of simplified versions of the previously known characterizations (co-NL and L) ;

— the obtention of characterizations of new classes such as regular languages and the class NL ;

— the definition of types corresponding to the classes characterized, a result which opens the way of using arguments based on testing.

These results, together with the results presented in the thesis, show that the GoI constructions are well-suited for the study of computational complexity. In particular, it provides a framework where both time and space complexity classes can be described and offers new methods and techniques for the study of those. Indeed, on one hand the characterizations of complexity classes as sets of operators thus obtained allows the use of tools and invariants of operator algebras. On the other hand, the obtention of these characterizations as types in the sense of GoI leads to the study of complexity classes by means of tests.

References

[AS12] Clément Aubert and Thomas Seiller. Characterizing co-NL by a group action. *Arxiv preprint arXiv :1209.3422*, 2012.

[AS13] Clément Aubert and Thomas Seiller. Logarithmic space and permutations. *Arxiv preprint*, abs/1301.3189, 2013.

[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.

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

[CFW81] Alain Connes, J. Feldman, and B. Weiss. An amenable equivalence relation is generated by a single transformation. *Ergodic Theory Dynamical Syst.*, 1(4) :431–450, 1981.

[Chu41] Alonzo Church. *The Calculi of Lambda Conversion.* Princeton University Press, Princeton, NJ, USA, 1941.

[Cur34] H. B. Curry. Functionality in combinatory logic. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 20, pages 584–590, November 1934.

[Dix54] Jacques Dixmier. Sous-anneaux abéliens maximaux dans les facteurs de type fini. *Annals of mathematics*, 59 :279–286, 1954.

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

[Ehr02] Thomas Ehrhard. On köthe sequence spaces and linear logic. *Mathematical Structures in Computer Science*, 12(5) :579–623, 2002.

[ER03] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309 :2003, 2003.

[ER06] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theoretical Computer Science*, 364(2) :166–195, 2006.

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

[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.

[Gen64] Gerhard Gentzen. Investigations into logical deduction. *American Philosophical Quarterly*, 1(4) :288–306, 1964.

[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) :1–102, 1987.

[Gir88a] J. Girard. Normal functors, power series and λ-calculus. *Annals of Pure and Applied Logic*, 37(2) :129–177, February 1988.

[Gir88b] Jean-Yves Girard. Geometry of interaction II : Deadlock-free algorithms. In *Proceedings of COLOG*, number 417 in Lecture Notes in Computer Science, pages 76–93. Springer, 1988.

[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.

[Gir95a] 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.

[Gir95b] Jean-Yves Girard. Light linear logic. In *Selected Papers from the International Workshop on Logical and Computational Complexity*, LCC '94, pages 145–176, London, UK, UK, 1995. Springer-Verlag.

[Gir06] Jean-Yves Girard. Geometry of interaction IV : the feedback equation. In Stoltenberg-Hansen and Väänänen, editors, *Logic Colloquium 2003*, pages 76 – 117. The Association for Symbolic Logic, 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.

[Har72] J. Hartmanis. On non-determinancy in simple computing devices. *Acta Informatica*, 1(4) :336–344, 1972.

[HKM08] Markus Holzer, Martin Kutrib, and Andreas Malcher. Multi-head finite automata : Characterizations, concepts and open problems. In Turlough Neary, Damien Woods, Anthony Karel Seda, and Niall Murphy, editors, *CSP*, volume 1 of *EPTCS*, pages 93–107, 2008.

[How80] William A. Howard. The formulas-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry : Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, 1980.

[Imm88] Neil Immerman. Nondeterministic space is closed under complementation. In *Structure in Complexity Theory Conference*, pages 112–115. IEEE, IEEE Computer Society, 1988.

[Laf04] Yves Lafont. Soft linear logic and polynomial time. *Theor. Comput. Sci.*, 318(1-2) :163–180, June 2004.

[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.

[Puk60] L. Pukanszky. On maximal albelian subrings of factors of type II_1. *Canad. J. Math.*, 12 :289–296, 1960.

[Sei12a] Thomas Seiller. Interaction graphs : Additives. *Arxiv preprint arXiv :1205.6557*, 2012.

[Sei12b] Thomas Seiller. Interaction graphs : Multiplicatives. *Annals of Pure and Applied Logic*, 163 :1808–1837, December 2012.

[Sei12c] Thomas Seiller. *Logique dans le facteur hyperfini : géometrie de l'interaction et complexité*. PhD thesis, Université de la Méditerranée, 2012.

[Sei13] Thomas Seiller. Interaction graphs : Exponentials. *Arxiv preprint arXiv :1312.1094*, 2013.

[Tur37] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1) :230–265, January 1937.