

INTERACTION GRAPHS AND COMPLEXITY I CLASSICAL AND PROBABILISTIC MULTIHEAD AUTOMATA

THOMAS SEILLER

Proofs, Programs, Systems (PPS). CNRS and Paris 7 University
Case 7014, 75205 Paris Cedex 13, France
e-mail address: seiller@pps.univ-paris-diderot.fr

ABSTRACT. The aim of Implicit Computational Complexity is to study algorithmic complexity only in terms of restrictions of languages and computational principles. Based on previous work about realisability models for linear logic, we proposed a new approach where semantical restrictions are considered instead of syntactical ones. This paper exposes the background and first results in this direction, showing the techniques lead to a hierarchies of models of (fragments of) linear logic mirroring subtle distinctions about computational principles. This is illustrated by the obtention of infinite families of distinct sublogarithmic complexity classes, for deterministic, non-deterministic as well as probabilistic computation. In particular, we obtain the first implicit characterisation of probabilistic logarithmic space predicates.

Draft version (compiled Wednesday 15th July, 2015 at 17:29).

1. INTRODUCTION

This paper is the first in a series which will provide new characterisations of complexity classes based on *complexity-through-realisation* techniques described in the perspective paper [Sei15b]. One of the final aims of this research program is to enable the use of techniques from the mathematical theories of operator algebras, dynamical systems and ergodic theory against open problems in complexity theory. We also believe that the *complexity-through-realisation* theory provides well-grounded mathematical foundations for complexity theory, one that is machine-independent and applies to different computational paradigms.

This paper exhibits a series of characterisations of sublogarithmic (predicates) complexity classes. It moreover lays ground to the general theory and provide the formal background and definitions. The author tried as much as possible to express these definitions in all their generality, so that they may be applied without being modified to study different computational paradigms such as quantum computation. The generality of the approach is illustrated

This work was partially supported by the the Marie Skłodowska-Curie Individual Fellowship (H2020-MSCA-IF-2014) 659920 - ReACT, as well as the French ANR project ANR-10-BLAN-0213 LOGOI and the french ANR project COQUAS (number 12 JS02 006 01).

by the obtention of first implicit characterisations of an infinite family of probabilistic complexity class, between stochastic languages STOCHASTIC and the class PLOGSPACE of (unbounded error) probabilistic logarithmic space. The interested reader can consult the complementary perspective paper [Sei15b] for an overview of the motivations and the general spirit behind this work.

Previous related work began in collaboration with C. Aubert [AS14a, AS14b] and provided new characterisations of logarithmic space complexity classes as sets of operators in a von Neumann algebra. The ideas behind this characterisations first appeared in a paper by Girard [Gir12] which suggested that one could characterise the class NLOGSPACE as a set of operators. Following Girard’s intuitions, we obtained two characterisations: CONLOGSPACE¹ [Sei12b, AS14a] and LOGSPACE [AS14b], although some of Girard’s definitions had to be modified to provide a complete proof of the characterisations. These modifications resulted in furthering this line of work from the initial setting, that is the geometry of interaction (GoI) models, i.e. specific realisability models for linear logic. As a consequence, although originating from considerations on logic, these results were not directly related to logic.

These works were then rephrased in more syntactic terms, providing new characterisations related to logic programming results [ABPS14, ABS15]. The approach taken in this paper is quite orthogonal to those results, since the latter are obtained by taking another step further from the initial framework of GoI models. After a first step which ended in the loss of an underlying logical framework, this second step ends in the loss of the mathematical foundations of the methods. Although this recent line of work have its own interests, it is the author’s belief that one should not forget the mathematical GoI models structure from which these characterisations originated. The present work therefore aims at obtaining a deeper understanding of how complexity classes can be related to the mathematical GoI models in order to provide complexity theorists with new techniques and invariants [Sei15b].

To achieve this goal, we build upon previous work by the author on the *geometry of interaction* (GoI) research program. The aim of the GoI program is to obtain particular kinds of realisability models for linear logic that accounts for the dynamics of cut-elimination – i.e. program execution through the Curry-Howard correspondence [Cur34, How80]. In a parallel series of papers [Sei12a, Sei14b, Sei14e], the author introduced – under the name *Interaction Graphs* – a family of models which generalises and axiomatizes the notion a GoI model. This systematic approach is based on the interpretation of programs as a generalisation of graphs named *graphings*. Graphings as considered by the author generalise a notion used to study measurable group actions [Ada90, Lev95, Gab00] which is in some ways a measurable variant of Cartan’s *pseudo-groups* [Car04, Car09, KN96]. The end result of this series of work is a very rich hierarchy of models parametrized by two monoids – the *weight monoid* Ω and a monoid of measurable maps m called a *microcosm* –, as well as a measurable map $m : \Omega \rightarrow \mathbf{R}_{\geq 0} \cup \{\infty\}$.

In some ways, the work presented here fills the gap between the above mentioned series of work on GoI models and on computational complexity. Indeed, we show how to obtain characterisations of CONLOGSPACE and LOGSPACE reformulated in the framework of interaction graphs. Those improve on earlier results using operator algebras [AS14a, AS14b] as they are now obtained as (meaningful) part of models of linear logic. The change of perspective from

¹Although it is known that CONLOGSPACE and NLOGSPACE are equal [Sze87, Imm88], the mentioned works willingly refrained from using this so-called Immerman-Szelepcsényi so that the developed methods may eventually lead to a new proof of this result.

this earlier approach furthermore extends the methods used for previous characterisations. As a consequence, we obtain characterisation of several classes that were not available using previous techniques, among which NLOGSPACE and the probabilistic complexity class PLOGSPACE.

Those characterisations of logarithmic space spaces predicates for distinct computational paradigms are obtained as parts of realisability model of linear logic corresponding to a given *weight monoid* and a given *microcosm*. It turns out that restrictions on the microcosm allows for a perfect correspondence between a part of the hierarchy of monoids of measurable maps and a part of the hierarchy of complexity classes. Indeed, we consider an infinite sequence $m_1 \subset m_2 \subset \dots \subset m_\infty$ of microcosms and show that the types of binary predicates in the associated models correspond to infinite sequences of (distinct) complexity classes between regular languages (resp. stochastic languages) and logarithmic space predicates. This result, together with the fact that the same monoid of measurable maps yields characterisations of the classes LOGSPACE, NLOGSPACE, CONLOGSPACE, and PLOGSPACE, supports the conjecture that there exists a correspondence between such monoids of measurable maps and complexity constraints [Sei15b].

1.1. Implicit Computational Complexity and Linear Logic. The field of Implicit Computational Complexity (ICC) aims at the obtention of characterisations of complexity classes without reference to specific machine models or external measure conditions. The objective in terms of restrictions on logical systems and/or on some computational principles. This effort has lead to several characterisations of the classes PTIME, PSPACE, LOGSPACE, etc. Among the results obtained in this domain, one can cite the constrained logical systems derived from linear logic. These systems are by-products of Girard's Linear Logic [Gir87a], which was introduced after a careful study of categorical models of lambda-calculus [Gir88]. Linear logic gives a logical status to the notion of resources, and allows a fine-tuned control over the potential re-use of resources though the introduction of the exponential connective. Indeed, linear logic is obtained through a decomposition of the intuitionistic implication $\Rightarrow$ in two independent operations: the exponential $!A$ which allows duplication, and a linear implication, written $\multimap$, which produces from a proof of A a proof of B , *consuming A in the process*. The implication $A \Rightarrow B$ is then obtained as the formula $!A \multimap B$. The re-use of resources being associated to the so-called exponential connective $!$, one may then modify the rules of this connective to obtain restrictions of linear logic which have a computational interest. This is how constrained linear logic systems, such as BLL [GSS92], SLL [Laf04] and LLL [Gir95b] which characterise PTime, or ELL [DJ03] which characterises Elementary time are obtained. Deciding if a program can be typed in such a system allows one to get static complexity bounds, and inference typed algorithms have thus been designed to this purpose.

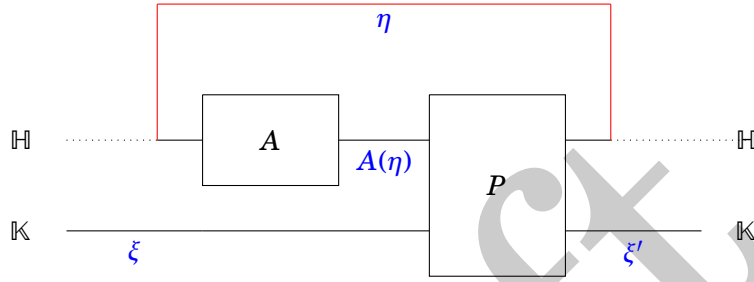
1.2. Geometry of Interaction. The *Geometry of Interaction* program was initiated by Girard and aims at obtaining a mathematical model of proofs and their normalization, contrarily to denotational semantics where a program given an argument is modeled by the same object as the result of the induced computation. In most of the proposed constructions of such models (called GoI models), programs and data are represented as operators acting on an infinite-dimensional (yet countable) Hilbert space. The execution of the program applied to an argument – a computation – then corresponds to solving an equation, called the *feedback equation* (illustrated in Figure 1). About ten years ago, Girard showed how this equation can be solved in a general setting [Gir06]. This lead him to the definition of a new construction of

$P \in \mathcal{L}(\mathbb{H} \oplus \mathbb{K})$ represents^a a program
 $A \in \mathcal{L}(\mathbb{H})$ represents an argument.
 $R \in \mathcal{L}(\mathbb{K})$ represents the result of the computation if:

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

^aHere, $\mathbb{H}$ and $\mathbb{K}$ are infinite (yet countable) dimensional Hilbert spaces, and $\mathcal{L}(\star)$ represents the set of operators acting on the Hilbert space Hilbert $\star$: bounded (iff continuous) linear maps from $\star$ to $\star$.

(a) Statement of the equation



(b) Illustration of the equation

Figure 1: The Feedback Equation

a geometry of interaction (GoI) in the hyperfinite factor [Gir11] – styled hyperfinite geometry of interaction in the following.

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 [Gir89] that 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].

More recent developments used techniques from operator algebras to define a GoI model where proofs are interpreted as operators in a type II von Neumann algebra [Gir11]. This model was shown to model the constrained linear logic ELL (Elementary Linear Logic), providing some hints of a correspondence between some (couples of) algebras and complexity classes [Sei14a]. This result was actually essential in the process of proposing the complexity-through-realisation methods explained in a perspective paper by the author [Sei15b] and applied in this paper. The “type II” GoI model also lead to characterisations of the classes CONLOGSPACE and LOGSPACE as sets of such operators [AS14a, AS14b]. Further work adapted these results in the framework of the more syntactic GoI model recently proposed by Girard [Gir13b, Gir13a], providing a bridge with logic programming [ABPS14, ABS15]. Although the logical aspects of the GoI model were lost in the process of obtaining these characterisations, these results are strongly related to the current paper.

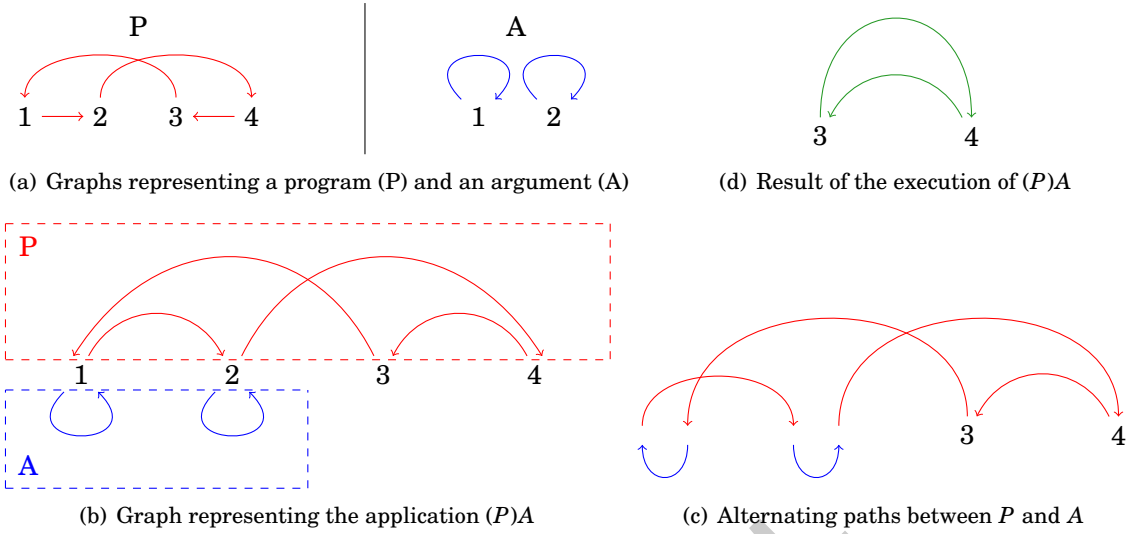


Figure 2: Model of the execution of programs (Interaction Graphs)

1.3. Interaction Graphs. In a series of papers [Sei12a, Sei14b], we introduced a combinatorial setting which both generalises and unifies the previously known geometry of interaction constructions introduced by Girard [Gir87b, Gir89, Gir95a, Gir11]. In this setting, programs are interpreted as graphs and the execution of programs is modeled by the construction of the graph of *alternating paths* between two given graphs (illustrated in Figure 2). What makes this approach particularly interesting is the fact that this construction yields the (necessarily unique) solution to the feedback equation (Figure 1). On top of this, the measurement used in Girard’s hyperfinite GoI can be computed as a measurement of a set of cycles [Sei12a]. This lead us to show that all known constructions of GoI were founded upon a unique geometric identity we called the *trefoil property* [Sei14b].

The main theoretical contributions of this work was that it put into light the trefoil property, a geometric identity relating sets of cycles and paths, which turned out to be the foundation of all GoI constructions. This property, which can be understood as a sort of associativity, suggests that computation – as modeled by geometry of interaction – is closely related to algebraic topology. From a more practical point of view, the objects under study are somehow finite representations of Girard’s infinite interpretations of proofs, and offer natural insights on an otherwise extremely complicated subject matter.

These constructions with graphs were then generalised. Even though the initial motivation for this generalisation was to deal with second-order quantification and exponential connectives, it quickly became clear to the author that one of the main interests of the obtained framework was in computational complexity. In this generalised setting, we showed [Sei14e] how to obtain a family of models which generalises and axiomatizes the notion a GoI model for the multiplicative-additive fragment. This systematic approach, extended by considering a generalisation of graphs named *graphings*, allows one to consider a very rich hierarchy of models parametrized, among other things, by two monoids: the *weight monoid* and a monoid of measurable maps called a *microcosm*. This was used to construct an interpretation of second-order quantification [Sei14e] and exponential connectives [Sei14d, Sei15a].

In this paper, we develop the first applications of this general framework in computational complexity (already sketched in previous papers [Sei14c, Sei15b]). We will explain how modifications of the two parameters – the weight monoid and the microcosm – allow for fine distinctions on the computational principles available in the model.

2. PRELIMINARIES: INTERACTION GRAPHS

This section provides the reader with needed definitions and results in order to understand the *complexity-through-realisability* approach described in the next section. Although self-contained, this reminder of previous work is not sufficient to gain all the needed intuitions to understand fully and in-depth the next sections. The reader familiar with the framework of interaction graphs can skip subsection 2.1 and subsection 2.2 as those consist in a very short reminder of previous papers [Sei12a, Sei14b, Sei14e]. In all cases, the subsection 4.1 describes a few new results which did not appear in previous work.

2.1. Graphs and Models. We briefly recall the main definitions and results that appeared in earlier papers [Sei12a, Sei14b, Sei14d] and which will be used later on.

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 $E^G \rightarrow \Omega$.

The construction of models of multiplicative-additive linear logic described in earlier work [Sei14b] relied on a single property, called the *trefoil property*, which relates the following 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 the set of cycles.

We recall below the exact definitions.

Definition 2. We define the following notions of cycles:

- a *cycle* in a graph F is a sequence $\pi = e_0 \dots e_n$ of edges such that for all $i > n$ the source of the edge e_{i+1} coincides with the target of the edge e_i ;
- a *1-cycle* in a graph F is a cycle π such that there are no cycle ρ and integer $k > 1$ with $\pi = \rho^k$, where ρ^k denotes the concatenation of k copies of ρ ;
- a *circuit* is an equivalence class of cycles for the equivalence relation defined by $e_0 \dots e_n \sim f_0 \dots f_n$ if and only if there exists an integer k such that for all i , $e_i = f_j$ with $j = k + i \bmod [n + 1]$.
- a *1-circuit* ρ is a circuit which is not a proper power of a smaller circuit, i.e. is the equivalence class of a 1-cycle.

Definition 3. Let S^G and D^G be finite sets. A *thick graph* G of carrier S^G and *dialect* D^G is a directed weighted graph over the set of vertices $S^G \times D^G$.

Remark 4. A good intuition for the set D^G is to understand it as a set of *control states*, as for instance in the definition of automata. As in the case of automata, changing the set of states through a bijection, i.e. renaming the states, intuitively does not change the thick graph. I.e. although described differently, the two graphs are virtually indistinguishable in the model. Such a renaming of a thick graph G is called a *variant* of G .

One can then define a notion of path, and a corresponding notion of cycles, following the latter intuition.

Definition 5. An alternating path π between two thick graphs F and G is a sequence $e_0 \dots e_n$ and an associated sequence $((v_i, d_i, d'_i))_{i=0}^{n+1}$ such that:

- $e_i \in E^F$ if and only if $e_{i+1} \in E^G$;
- for all $i = 0, \dots, n$, if $e_i \in E^F$, then $s^F(e_i) = (v_i, d_i)$, $t^F(e_i) = (v_{i+1}, d_{i+1})$ and $d'_i = d'_{i+1}$;
- for all $i = 0, \dots, n$, if $e_i \in E^G$, then $s^G(e_i) = (v_i, d'_i)$, $t^G(e_i) = (v_{i+1}, d'_{i+1})$ and $d_i = d_{i+1}$;

We write $\text{Path}(F, G)$ the set of alternating paths between F and G , and denote by $\text{Path}(F, G)_V$ the subset of $\text{Path}(F, G)$ consisting of those paths whose source (v_0, d_0, d'_0) and target $(v_{n+1}, d_{n+1}, d'_{n+1})$ satisfy $v_0 \in V$ and $v_{n+1} \in V$.

Definition 6. The execution $F :: G$ of two thick graphs F, G is the thick graph of carrier $S^F \Delta S^G$ – the symmetric difference between S^F and S^G – and dialect $D^F \times D^G$ defined by $E^{F :: G} = \text{Path}(F, G)_{S^F \Delta S^G}$ with obvious source and target maps. The weight map is defined so that the weight of a path $\pi = e_0 \dots e_n$ is the product of the weights of the edges e_i .

Definition 7. An *alternating cycle* between F and G is an alternating path $\pi = e_0 \dots e_n$ whose source is equal to its target and such that $e_0 \in E^F$ if and only if $e_n \in E^G$.

An *alternating circuit* is an equivalence class of alternating cycles for the equivalence relation² defined by $e_0 \dots e_n \sim f_0 \dots f_n$ if and only if there exists an integer k such that for all i , $e_i = f_j$ with $j = k + i \mod [n + 1]$.

An *alternating 1-circuit* ρ is an alternating circuit which is not a proper power of a smaller circuit, i.e. there does not exist an alternating circuit ρ and an integer $k > 1$ such that $\pi = \rho^k$, where the power denotes iterated concatenation.

Definition 8. Let F and G be two thick graphs. We define $\text{Circ}(F, G)$ as the set of alternating 1-circuits between F and G .

Definition 9. Being given a map $m : \Omega \rightarrow \mathbf{R}_+ \cup \{\infty\}$, one can define a measurement of the interaction between thick graphs. For every couple of thick graphs F, G , it is defined as:

$$\llbracket F, G \rrbracket_m = \sum_{\pi \in \text{Circ}(F, G)} \frac{1}{\text{Card}(D^F \times D^G)} m(\omega(\pi))$$

In order to define additive connectives, one has to consider (weighted) sums of (thick) graphs [Sei14b].

Definition 10. A *sliced and thick* graph G of carrier V^G is a finite formal weighted sum $\sum_{i \in I^G} \alpha_i^G G_i$ where I^G is the index set of G , and for each $i \in I^G$, α_i^G is a real number and G_i is a sliced graph of carrier V^G . We define $\mathbf{1}_G = \sum_{i \in I^G} \alpha_i^G$.

A thick and sliced graph is *slice-free* when I^G is a singleton set $\{i\}$ and $\alpha_i^G = 1$. It is *dialect-free* when it is slice-free and moreover the dialect of A_i is a singleton set.

The measurement and execution defined above are extended in a straightforward way to sliced and thick graphs.

$$\llbracket \sum_{i \in I^G} \alpha_i^G G_i, \sum_{j \in I^H} \alpha_j^H H_j \rrbracket_m = \sum_{i \in I^G} \sum_{j \in I^H} \alpha_i^G \alpha_j^H \llbracket G_i, H_j \rrbracket_m$$

²Notice that the condition that e_0 and e_n do not belong to the same graph is needed for the cyclic permutation of an alternating cycle to be an alternating cycle itself.

$$\left(\sum_{i \in I^G} \alpha_i^G G_i \right) :: \left(\sum_{j \in I^H} \alpha_j^H H_j \right) = \sum_{i \in I^G} \sum_{j \in I^H} \alpha_i^G \alpha_j^H G_i :: H_j$$

For technical reasons explained in earlier work [Sei12a], one has to work with couples (a, A) – named *projects* – of a real number a together with a sliced and thick graph A . From the previous measurement, one can define an orthogonality relation between projects (a, A) and (b, B) :

$$(a, A) \perp (b, B) \text{ if and only if } a + b + \llbracket A, B \rrbracket_m \neq 0, \infty$$

From these notions of *projects* and *orthogonality*, one can define objects generalising the notion of type, called *conducts*. A conduct is simply a set of projects which is equal to its bi-orthogonal closure or, equivalently, a set $\mathbf{A}$ of projects such that there exists a set T_A of projects – intuitively a set of tests – such that $\mathbf{A} = T_A^\perp$. This notion of type is quite natural and reminiscent of work on classical realisability [Kri01, Rib07, Kri09].

One can then show that conducts³ can be arranged so as to form a model of multiplicative-additive linear logic, by producing constructs on conducts that interpret linear logic connectives. The details of this construction are given in our earlier paper on additives [Sei14b], whose main result can now be stated.

Theorem 11. *Let Ω be a monoid and m be any map $\Omega \rightarrow \mathbf{R}_{\geq 0}$. The set of Ω -weighted thick and sliced graphs with the orthogonality defined from m provides a model of Multiplicative-Additive Linear Logic (MALL).*

2.2. Graphings. The finite nature of graphs imposes a limit on the expressivity of the models. In particular if one wants to define exponential connectives, one has to use infinite objects. This can be done in a discrete way by considering infinite graphs, but one can also consider continuous models in which the graphs act on infinite points but keep some properties of finite graphs thanks to measure theory. This second solution is more involved than the first one and lead the author to work with the notion of *graphing* – a realisation of a graph on a measure space – which has been used and studied in the context of ergodic theory and operator algebras [Ada90, Lev95, Gab00]. This solution turns out to be the right generalisation as the usual notions of graphs – finite or infinite – can be recovered as particular kinds of graphings, namely graphings over discrete spaces with counting measure.

The notion of graphing studied in earlier work [Sei14e] is parametrized by the choice of a monoid of measurable maps which are the possible realisations of edges. Taking into account this monoid of measurable maps – called a *microcosm* – allows for a fine-grained analysis of the families of models one can define in this way. This notion of microcosm will be most important in this paper as it will be shown to relate to complexity constraints.

Definition 12. Let $\mathbf{X} = (X, \mathcal{B}, \lambda)$ be a measure space. We denote by $\mathcal{M}(\mathbf{X})$ the set of Borel-preserving non-singular transformations⁴ $X \rightarrow X$. A *microcosm* of the measure space X is a subset $\mathfrak{m}$ of $\mathcal{M}(\mathbf{X})$ which is closed under composition and contains the identity.

³To be more precise, one need to consider a refined notion of conducts called *behaviors*. We refer to our paper on additives [Sei14b] or subsection 2.3 for more details.

⁴A non-singular transformation $f : X \rightarrow X$ is a measurable map which preserves the sets of null measure, i.e. $\lambda(f(A)) = 0$ if and only if $\lambda(A) = 0$. The map $f : X \rightarrow X$ is *Borel-preserving* when the image of a Borel set by f is still a Borel set.

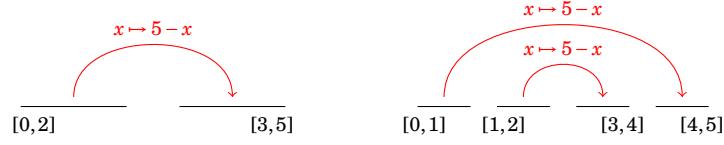


Figure 3: A graphing on the real line and one of its refinements

Remark 13. This definition of microcosms is the original notion exposed in our work introducing graphings [Sei14e]. It must be pointed out, however, that it was generalised in order to deal with continuous dialects and interpret second-order linear logic with “full” exponentials [Sei15a]. Since we will focus here on discrete dialects, we deemed unnecessary to expose the more general – and more complicated – setting.

We now 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 14 (Graphings). Let $\mathfrak{m}$ be a microcosm of a measure space $\mathbf{X} = (X, \mathcal{B}, \lambda)$ and V^F a measurable subset of X . A Ω -*weighted graphing* in $\mathfrak{m}$ of carrier V^F is a countable family $F = \{(\omega_e^F, \phi_e^F : S_e^F \rightarrow T_e^F)\}_{e \in E^F}$, 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 ;
- ϕ_e^F is an element of $\mathfrak{m}$, the *realisation* 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 .

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 $e_i \in G'$ ($i = 1, \dots, n$) 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 $i = 1, \dots, n$, $\phi_{e_i}^{G'}$ is equal to ϕ_e^G .

Such a graphing G' is an example of a *refinement* of G , and one can easily generalise the previous conditions to define a general notion of refinement of graphings. Figure 3 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 end result is that a graphing is only a presentation – as a sort of graph – of a generalised measured dynamical system. As such, two graphings F and G are identified as long as they represent the same dynamics, i.e. if for almost every $x \in X$, the multisets $Gx = \{y \in X \mid \exists e \in E^G, \phi_e^G(x) = y\}$ and $Hx = \{y \in X \mid \exists e \in E^H, \phi_e^H(x) = y\}$ are equal.

The definitions of execution and measurement recalled above 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 $\mathfrak{m}$ ⁵. A more involved argument [Sei14e] then shows that the trefoil property holds for a family of measurements

⁵As a consequence, a microcosm is a closed world for the execution of programs.

$\llbracket \cdot, \cdot \rrbracket_m$, where $m : \Omega \rightarrow \mathbf{R}_{\geq 0} \cup \{\infty\}$ is any measurable map. These results [Sei14e] are obtained as a generalisation of constructions considered in the author's thesis⁶ on directed graphs and lead to the main theorem which states that for a given set of parameters, the corresponding notion of graphings defines a model of Multiplicative-Additive Linear Logic.

Theorem 15 (Seiller [Sei14e]). *Let Ω be a monoid, $\mathfrak{m}$ a microcosm. For every choice of measurable map $m : \Omega \rightarrow \mathbf{R}_{\geq 0} \cup \{\infty\}$, the set of Ω -weighted graphings in $\mathfrak{m}$ yields a model of Multiplicative-Additive Linear Logic (MALL).*

We will formally state a refinement of this main theorem in the following, but we first provide the reader with a quick sketch of the construction of the models.

2.3. Models from graphings: formally. One generalises naturally the notion of graphing to deal with a notion of states. Indeed, the dialect is a (finite and discrete) measure space when considered with the counting measure, leading to a notion of “graphing with states” – styled *thick graphings* – for which notion of paths and cycles are straightforwardly adapted from their counterpart on graphs. Then, one can consider formal weighted sums of thick graphings as we explained in the setting of graphs to define the notion of *sliced and thick graphings* [Sei14d]. In most of the models, one can define exponential modalities. A first example of a model that can interpret Elementary Linear Logic [Gir95b] – a logical system capturing elementary time computable functions – was obtained by the author [Sei12b, Sei14d]. Recently, the author provided a model of full linear logic based on an extension of this setting to *continuous dialects* [Sei15a]. We will here define models in which one can define some sorts of exponential connectives, even though these do not satisfy the usual properties of exponentials in linear logic or elementary linear logic.

From now on, the term *graphing* will be used as a substitute for *thick and sliced graphing* and the term *project* will be used to denote couples (a, A) where a is a real number and A a (thick and sliced) graphing. Projects are in some ways generalisations of proofs. Indeed, we interpret proofs as projects in the resulting model, but not all projects come from a proof!

Definition 16 (Projects). A *project* of carrier V is a couple $\mathfrak{a} = (a, A)$ consisting of an element a of $\mathbf{R} \cup \{\infty\}$ and a thick and sliced graphing A of carrier $V^A \subset V$.

For a sliced graphing $A = \sum_{i \in I^A} \alpha_i^A A_i$, we write $\mathbf{1}_A$ the sum $\sum_{i \in I^A} \alpha_i^A$. We say the graphing is *slice-free* when I^A is a singleton set $\{u\}$ and $\alpha_u^A = 1$. It is *dialect-free* when it is slice-free and the dialect of A_u is a singleton set.

Following the technique explained above in the simpler setting on graphs, we define a measurement between graphings and define from it a notion of orthogonality. This measurement between two graphings F and G is defined by measuring alternating cycles – more precisely alternating 1-circuits which are specific alternating cycles – between F and G . We refer to previous work for the formal definition of alternating 1-circuits in the context of graphings [Sei14e], and write $\text{Circ}^m(F, G)$ the set of alternating 1-circuits between the graphs F, G . Each such circuit π is realised by a map $\phi_\pi : X \rightarrow X$ in $\mathfrak{m}$ and has a *support*, i.e. a measurable subset $\text{supp}(\pi)$ such that $\phi_\pi(\text{supp}(\pi)) = \text{supp}(\pi)$. We then define, for all measurable maps $\Omega \rightarrow \mathbf{R}_{\geq 0} \cup \{\infty\}$, a measurement $\llbracket F, G \rrbracket_m$ between F and G . For the sake of simplicity, and because we won't face a more complex situation in this paper, we give the explicit expression

⁶In the cited work, the results were stated in the particular case of the microcosm of measure-preserving maps on the real line.

of this measurement in the restricted case when all elements of $\mathbf{m}$ are measure-preserving. The measurement in this case can be defined as

$$\llbracket F, G \rrbracket_{\mathbf{m}} = \sum_{\pi \in \text{Circ}^{\mathbf{m}}(F, G)} \int_{\text{supp}(\pi)} \frac{m(\omega(\pi)^{\rho_{\phi_{\pi}}(x)})}{\rho_{\phi_{\pi}}(x)} d\lambda(x)$$

with ρ_{ϕ} the map $X \rightarrow \mathbf{N} \cup \{\infty\}$ defined as $x \mapsto \inf\{n \in \mathbf{N} \cup \{\infty\} \mid \phi^n(x) = x\}$.

Definition 17 (Orthogonality). Two projects $\mathbf{a} = (a, A)$ and $\mathbf{b} = (b, B)$ are *orthogonal*, denoted $\mathbf{a} \perp \mathbf{b}$, when $\llbracket \mathbf{a}, \mathbf{b} \rrbracket_{\mathbf{m}} \neq 0, \infty$, where $\llbracket \mathbf{a}, \mathbf{b} \rrbracket_{\mathbf{m}}$ is defined as the following expression.

$$\llbracket \mathbf{a}, \mathbf{b} \rrbracket_{\mathbf{m}} = a \mathbf{1}_B + b \mathbf{1}_A + \llbracket A, B \rrbracket_{\mathbf{m}}$$

Given a set of projects E , we define its orthogonal $E^{\perp}$ as the set $\{\mathbf{f} \mid \forall \mathbf{e} \in E, \mathbf{e} \perp \mathbf{f}\}$.

This notion of orthogonality is then used to define *conducts*, which are used to interpret types in the model. Intuitively, the orthogonality represents linear negation at the level of projects.

Definition 18 (Conducts). A conduct $\mathbf{A}$ of carrier V is a set of projects of carrier V which is equal to its bi-orthogonal closure, i.e. $\mathbf{A} = \mathbf{A}^{\perp\perp}$.

It is a known fact (but usually not mentionned) that this definition is completely equivalent to the following one:

A conduct of carrier V is a set $\mathbf{A}$ of projects of carrier V such that there exists a set B of projects with $\mathbf{A} = B^{\perp}$.

We believe this alternative definition to be more intuitive, especially for the reader familiar with the theory of proof nets for linear logic. We refer the reader to a work by Naibo, Petrolo and the author [NPS15] for a detailed discussion of the ties between orthogonality and the correctness criterions for proof nets.

The execution of projects is then defined based on this measurement and the operation of *execution of graphings*. The execution $F :: G$ between two graphings F, G of respective carriers V^F and V^G is defined as the graphing of all restrictions of alternating paths between F and G so that the sources and targets are subsets of the symmetric difference $V^F \Delta V^G$. The complete definition can be found in earlier work [Sei14e]. This notion of execution is extended to define the execution between projects as follows.

Definition 19 (Execution). Let $\mathbf{a} = (a, A)$ and $\mathbf{b} = (b, B)$ be two projects of respective carriers $V + W$ and $V + Z$ (V, W, Z are pairwise disjoint). Their execution is the project $\mathbf{a} :: \mathbf{b} = (\llbracket \mathbf{a}, \mathbf{b} \rrbracket_{\mathbf{m}}, A :: B)$ of carrier $W + Z$.

As a special case of execution, one obtains the definition of the tensor product of two projects $\mathbf{f} = (f, F)$ and $\mathbf{g} = (g, G)$: if V^F and V^G are disjoint (i.e. their intersection is negligible), their execution is written $\mathbf{f} \otimes \mathbf{g} = (f \mathbf{1}_G + g \mathbf{1}_F, F \otimes G)$. In that case, notice that all alternating paths are of length 1 and the result looks like a disjoint union⁷.

The additional structure provided by *slices* allows for another construction on projects.

Definition 20 (Sum). Let $\mathbf{a} = (a, A)$ and $\mathbf{b} = (b, B)$ be two projects of respective dialects D^A and D^B . We define their sum $\mathbf{a} + \mathbf{b}$ as the project $(a + b, A + B)$ of dialect $D^A + D^B$.

⁷It is not a disjoint union because of dialect – states. However, if both graphings are slice-free the graphings of the tensor product $F \otimes G$ is exactly the disjoint union of F and G .

Notations 21. In the following, we will denote by o_V the project $(0, \emptyset)$ of carrier V , where $\emptyset$ denotes the empty graphing. We also denote by 1_V the project $(0, \text{Id}_V)$ of carrier V , where Id_V denotes the identity graphing, i.e. the graphing with a single edge of weight 1, source and target V , and realised by the identity.

The previous definition allows for the interpretation of additive connectives. Given two projects a and b , we define $a \& b$ as the project $a \otimes o_{VB} + b \otimes o_{VA}$. The interpretation of the $\oplus$ rules which builds a proof of $A \otimes B$ from a proof of A is obtained by constructing, from a project a , the project $a \otimes o_{VB}$.

Now, if the interpretation of multiplicative connectives works well in the general setting of conducts, one need to restrict to so-called *behaviors* to obtain a sound representation of additive connectives.

Definition 22 (Behaviors). A behavior $\mathbf{B}$ of carrier V is a conduct such that for all $\lambda \in \mathbf{R}$:

- if $b \in \mathbf{B}$, then $b + \lambda o_V \in \mathbf{B}$;
- if $b \in \mathbf{B}^\perp$, then $b + \lambda o_V \in \mathbf{B}^\perp$;

To gain some intuitions, let us say that if conducts are types, then behaviors are some sort of *strict linear types*. Indeed, the set of behaviors form a model of multiplicative-additive linear logic that do not validate any of the structural rules. Let us now define the interpretations of linear logic connectives at the level of types.

Definition 23 (Multiplicatives). Given two conducts $\mathbf{A}, \mathbf{B}$ of disjoint carriers, we define:

$$\begin{aligned} \mathbf{A} \otimes \mathbf{B} &= \{a \otimes b \mid a \in \mathbf{A}, b \in \mathbf{B}\}^{\perp\perp} \\ \mathbf{A} \multimap \mathbf{B} &= \{f \mid \forall a \in \mathbf{A}, f :: a \in \mathbf{B}\} \\ \mathbf{A} \& \mathbf{B} &= \{a' \otimes o_{VB} \mid a' \in \mathbf{A}^\perp\}^\perp \cap \{b' \otimes o_{VA} \mid b' \in \mathbf{B}^\perp\}^\perp \\ \mathbf{A} \oplus \mathbf{B} &= (\{a \otimes o_{VB} \mid a \in \mathbf{A}\} \cup \{b \otimes o_{VA} \mid b \in \mathbf{B}\})^{\perp\perp} \end{aligned}$$

As the notion of conduct generalises that of type, we will in this paper allow ourselves to use the more usual term “types” to refer to conducts, may they be behaviors or not. When needed, we will however use the terms *conduct* or *behavior* to be more specific. In particular, as behaviors are specific conducts, we will use the term conduct exclusively for those types which are not behaviors.

2.4. A General Model of Computation. We will here explain how graphings provide a satisfactory mathematical abstraction of the notion of computation. In other words, we will motivate the working basis of our approach to complexity, namely the slogan “Programs as Graphings”. Since Girard’s *Geometry of Interaction* constructions appear as special cases of our own Interaction Graphs construction, it is possible to adapt Danos’ interpretation [Dan90] of pure lambda-calculus in geometry of interaction. As a consequence, the notion of graphing is expressive enough when considering classical, deterministic, sequential, computation. Moreover, graphings can represent non-deterministic and probabilistic computation, as we will explain below. A representation of quantum computation in terms of graphing should not raise major difficulties since the theory of graphings is strongly related to functional analysis and linear algebra. Lastly, we believe that both concurrent computation and more abstract models such as cellular automata could be represented in this setting as well, since graphings are in some ways generalisations of measured dynamical systems⁸.

⁸A notion of *topological graphings* which is a kind of generalised topological dynamical system can also be considered [Sei14e] although we chose to focus on the measurable version for now.

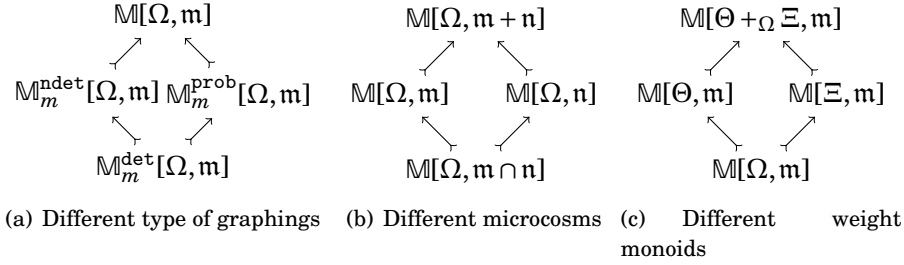


Figure 4: Inclusions of models

Moreover, all these notions of computation can be represented in the same model. Indeed, we can consider particular kinds of graphings, e.g. probabilistic graphings, that represent particular ways of computing. These particular kinds of graphings are shown to define a model of computation, e.g. if one composes two probabilistic graphings, the result is a probabilistic graphing. However, a deterministic graphing is a special kind of probabilistic graphing, and the deterministic model is therefore embedded into the probabilistic model. Similarly, there exists a model in which both probabilistic and non-deterministic graphings coexist (thus containing some non-deterministic *and* probabilistic graphings). In general, a model is then described by providing:

- a microcosm m ;
- a weight monoid Ω ;
- a map $m : \Omega \rightarrow \mathbf{R}_{\geq 0} \cup \{\infty\}$;
- a restriction on graphings.

The general model of all graphings (without restrictions) contains all others and is denoted by $M_m[\Omega, m]$. Then the deterministic, non-deterministic and probabilistic submodels $M_m^{\det}[\Omega, m]$, $M_m^{\text{ndet}}[\Omega, m]$ and $M_m^{\text{prob}}[\Omega, m]$ are naturally embedded into this general model (4(a)). But these are not the only embeddings one can consider. Indeed, if a microcosm m is contained in a microcosm n then a m -graphing is also a n -graphing. If we denote, in general, by $n+m$ the *smallest common extension* of n and m , and by $n \cap m$ the intersection of n and m , one obtains the embeddings described in 4(b). Lastly, the same sort of phenomenon appears from considering variations on the weight monoid. Writing Ω , Θ and Ξ three monoids with inclusions $\Xi \rightarrow \Omega$ and $\Xi \rightarrow \Theta$, one can define the amalgamated sum $\Theta +_{\Omega} \Xi$ and obtain the natural embeddings shown in 4(c).

We now define the notions of *deterministic*, *non-deterministic*, and *probabilistic* graphings. Other restrictions on graphings (e.g. quantum graphings) could be considered but we reserve their study for forthcoming works. We write 1 the unit of the weight monoid Ω .

Definition 24. A graphing G is deterministic if:

- for all $e \in E^G$, $\omega^G(e) = 1$;
- the set $\{x \in X \mid \text{Card}(\{e \in E^F \mid x \in S_e^G\}) \geq 2\}$ is of null measure.

A project (a, A) is deterministic if A is deterministic.

Theorem 25. If G, H are deterministic graphings, their execution $G :: H$ is deterministic.

Proof. A deterministic graphing F satisfies that for every edges $e, f \in E^F$, $S_e^F \cap S_f^F$ is of null measure. Suppose that the graphing $F :: G$ is not deterministic. Then there exists a Borel B of non-zero measure and two edges $e, f \in E^{F :: G}$ such that $B \subset S_e^{F :: G} \cap S_f^{F :: G}$. The edges e, f

correspond to paths π_e and π_f alternating between F and G . It is clear that the first step of these paths belong to the same graphing, say F without loss of generality, because the Borel set B did not belong to the *cut*. Thus π_e and π_f can be written $\pi_e = f_0\pi_e^1$ and $\pi_f = f_0\pi_f^1$. Thus the domains of the paths π_e^1 and π_f^1 coincide on the Borel set $\phi_{f_0}^F(B)$ which is of non-zero measure since all maps considered are non-singular. One can then continue the reasoning up to the end of one of the paths and show that they are equal up to this point. Now, if one of the paths ends before the other we have a contradiction because it would mean the the Borel set under consideration would be at the same time inside and outside the cut, which is not possible. So both paths have the same length and are therefore equal. Which shows that $F :: G$ is deterministic since we have shown that if the domain of two paths alternating between F and G coincide on a non-zero measure Borel set, the two paths are equal (hence they correspond to the same edge in $F :: G$). $\square$

Of course, the notion of (*classical*) *non-deterministic graphing* is obtained by simply overlooking the second condition in the definition of deterministic graphings.

Definition 26. A graphing G is non-deterministic if:

- for all $e \in E^G$, $\omega^G(e) = 1$;

A project (a, A) is non-deterministic if A is non-deterministic.

We state the closure under execution of the set of non-deterministic graphings, but omit the proof as it is completely straightforward.

Theorem 27. *If G, H are non-deterministic graphings, their execution $G :: H$ is a non-deterministic graphing.*

For the following definition (probabilistic graphings), we suppose that the weight monoid Ω comes equipped with an injective morphism $[0, 1] \rightarrow \Omega$ where $[0, 1]$ denotes the unit segment with its natural monoid structure, i.e. with multiplication. We abusively denote by $[0, 1]$ the image of this morphism.

Definition 28. A graphing F is probabilistic if:

- for all $e \in E^F$, $\omega^F(e) \in [0, 1]$;
- the set $\text{Out}(F) = \{x \in X \mid \sum_{\{e \in E^F \mid x \in S_e^F\}} \omega^F(e) > 1\}$ is of null measure.

A project (a, A) is probabilistic if A is probabilistic.

Theorem 29. *If G, H are probabilistic graphings, their execution $G :: H$ is probabilistic.*

Proof. If the weights of edges in F and G are elements of $[0, 1]$, then it is clear that the weights of edges in $F :: G$ are also elements of $[0, 1]$. We therefore only need to check that the second condition is preserved.

Let us denote by $\text{Out}(F :: G)$ the set of $x \in X$ which are source of paths whose added weight is greater than 1, and by $\text{Out}(F \cup G)$ the set of x which are source of edges (either in F or G) whose added weight is greater than 1. First, we notice that if $x \in \text{Out}(F :: G)$ then either $x \in \text{Out}(F \cup G)$, or x is mapped – through at least one edge – to an element y which is itself in $\text{Out}(F \cup G)$. To prove this statement, let us write $\text{paths}(x)$ (resp. $\text{edges}(x)$) the set of paths in $F :: G$ (resp. edges in F or G) whose source contain x . We know the sum of all the weights of these paths is greater than 1, i.e. $\sum_{\pi \in \text{paths}(x)} \omega(\pi) > 1$. But this sum can be rearranged by ordering paths depending on their initial edge, i.e. $\sum_{\pi \in \text{paths}(x)} \omega(\pi) = \sum_{e \in \text{edges}(x)} \sum_{\pi = e\rho \in \text{paths}(x)^e} \omega(\pi)$, where $\text{paths}(x)^e$ denotes the paths whose first edge is e . Now,

since the weight of e appears in all $\omega(e\rho) = \omega(e)\omega(\rho)$, we can factorize and obtain the following inequality.

$$\sum_{e \in \text{edges}(x)} \omega(e) \left(\sum_{\pi=e\rho \in \text{paths}(x)^e} \omega(\rho) \right) > 1$$

Since the sum $\sum_{e \in \text{edges}(x)} \omega(e)$ is not greater than 1, we deduce that there exists at least one $e \in \text{edges}(x)$ such that $\sum_{\pi=e\rho \in \text{paths}(x)^e} \omega(\rho) > 1$. However, this means that $\phi_e(x)$ is an element of $\text{Out}(F :: G)$.

Now, we must note that x is not element of a cycle. This is clear from the fact that x lies in the carrier of $F :: G$.

Then, an induction shows that x is an element of $\text{Out}(F :: G)$ if and only if there is a (finite, possibly empty) path from x to an element of $\text{Out}(F \cup G)$, i.e. $\text{Out}(F :: G)$ is at most a countable union of images of the set $\text{Out}(F \cup G)$. But since all maps considered are non-singular, these images of $\text{Out}(F \cup G)$ are negligible subsets since $\text{Out}(F \cup G)$ is itself negligible. This ends the proof as a countable union of copies of negligible sets are negligible (by countable additivity), hence $\text{Out}(F :: G)$ is negligible. $\square$

The previous theorems show that the restrictions on graphings that will be considered in this paper are preserved by execution. We can then follow the construction of models described in the previous section. Since the proofs that the obtained notion of type provides a model of MALL make use of deterministic graphings only, i.e. graphings that belong to all of the models considered, the following result is obtained by a straightforward adaptation of the proof of Theorem 15.

Definition 30. Let Ω be a monoid, $\mathfrak{m}$ a microcosm, m a measurable map $\Omega \rightarrow \mathbf{R}_{\geq 0} \cup \{\infty\}$, and $\mathfrak{t} \in \{\text{all}, \text{det}, \text{ndet}, \text{prob}\}$ a type of graphings. We denote by $\mathbb{M}_m^{\mathfrak{t}}[\Omega, \mathfrak{m}]$ the model built on the set of Ω -weighted thick $\mathfrak{t}$ -graphings in $\mathfrak{m}$ with the orthogonality parametrized by m .

Theorem 31. We keep the notations of the previous definition. The model $\mathbb{M}_m^{\mathfrak{t}}[\Omega, \mathfrak{m}]$ soundly interprets MALL.

3. THEORY OF $(\Omega, \mathfrak{m})$ -MACHINES

We describe in this section how the models described above provide all the necessary tools for defining complexity classes. For this, we first provide a very general theory for exponential connectives. Then, we explain how words over a given alphabet can be represented in the models; we thus define the type of words and consider *(predicate) machines* as specific elements of the type $!\mathbf{Words}_{\Sigma}^{(2)} \multimap \mathbf{NBool}$, where $\mathbf{Words}_{\Sigma}^{(2)}$ is the type of words over the alphabet Σ and $\mathbf{NBool}$ is in some ways a type of *non-deterministic booleans*. Then, we explain how these notions can be used to represent computation: we introduce the notion of test, and explain how a machine accepts an input word w.r.t. a chosen test.

3.1. Exponential Connectives. As explained in earlier papers [Sei14d, Sei15a], exponential connectives consist in a quite (intuitively) simple operation on graph(ing)s: from a thick graph(ing), define a dialect-free graph(ing). The main challenge is to do so without losing any information in the process. We here introduce the notion of *perennial situation* on a space $\mathbf{X}$, which corresponds to the description of a canonical n -fold decomposition for all integer n . Intuitively, this canonical n -fold decomposition $(\mathbf{X}_i)_{i=0}^n$ can then be used to encode the discrete dialect by pre- and post-composing edges of a graphing G with chosen Borel automorphisms

between $\mathbf{X}$ and the splittings $\mathbf{X}_i$. The choice of such a family of Borel automorphism is named an *exponential structure* (Definition 36). This gives the intuitions behind the most simple exponential connectives one can define. To be as general as possible, we actually introduce a framework where exponentials can be defined through families of perennial situations.

Definition 32 (Perennial Situation). A *perennial situation* P on a measure space $\mathbf{X} = (X, \mathcal{B}, \lambda)$ is a choice, for all integer $n \in \mathbf{N}$, of a family of measurable subspaces $X^n(0), \dots, X^n(n)$ of X such that:

- $\cup_{i=0}^n X^n(i) = X$;
- $\lambda(X^n(i) \cap X^n(j)) = 0$ if $i \neq j$;
- there exists bi-measurable non-singular transformations $t_p^n[i, j] : X^n(i) \rightarrow X^n(j)$ such that $t_p^n[j, k] \circ t_p^n[i, j] = t_p^n[i, k]$;

Remark 33. As an example of a perennial situation, consider the space $[0, 1]$ with Lebesgue measure. It can be cut into $n + 1$ pieces for all integer n as follows: $X^n(i) = [\frac{i}{n+1}, \frac{i+1}{n+1}]$. The canonical bijections are then defined as the translations.

Of course, not all measure space admit a perennial situation. The term *perennial space* will be used for those space which admit a perennial situation. Moreover a given space, say $[0, 1]$, can be endowed with many perennial situations: the image of a perennial situation through any Borel automorphism of the space is a perennial situation. We will now consider spaces endowed with a set of perennial situations – we call this an *exponential space*; for each graphing G , each perennial situations will be used to define an exponentiation of G . In order to define exponentials correctly, though, one needs to ensure that the splitting induced by the perennial situation is coherent with the chosen microcosm. This additional structure is contained in the smallest perennial microcosm $\text{per}_{\mathbf{X}}$ over an exponential space $\mathbf{X}$ – defined below.

Definition 34 (Exponential Space). An exponential space is a measure space $\mathbf{X}$ together with a family $\text{Exp}(\mathbf{X})$ of perennial structures over $\mathbf{X}$.

Definition 35 (Perennial Microcosm). Let $\mathbf{X}$ be an exponential space. A microcosm $\mathfrak{m}$ over $\mathbf{X}$ is *perennial* when for all perennial structure P in $\text{Exp}(\mathbf{X})$, and all integers n, i, j , the canonical bijection $t_p^n[i, j]$ is the restriction of an element $T \in \mathfrak{m}$ to the subspace $X^n_P(i)$. We write $\text{per}_{\mathbf{X}}$ the smallest perennial microcosm over $\mathbf{X}$.

Definition 36 (Exponential Structure). An exponential structure on an exponential space $\mathbf{X}$ is given by, for all perennial structure P and all integer n , a family $((B_p^\mu[n, i])_{0 \leq i \leq n})_{\mu \in I_n}$ of families of Borel automorphisms, where the I_n are index sets, and:

- **(Compatibility)** for every natural number $n \geq 1$ and $\mu \in I_n$, $B_p^\mu[n, i]$ is a Borel isomorphism $X \rightarrow X[n, i] \subset X^n_P(i)$ such that $B_p^\mu[n, j] \circ B_p^\mu[n, i]^{-1}$ lies in the smallest perennial microcosm $\text{per}_{\mathbf{X}}$;
- **(Symmetry)** for every natural number $n \geq 1$, every $\mu \in I_n$, and every permutation σ over the set $\{1, \dots, n\}$, there exists $\nu \in I_n$ such that $B_p^\mu[n, i] = B_p^\nu[n, \sigma(i)]$.

An exponential space $(\mathbf{X}, \text{Exp}(\mathbf{X}))$ together with an exponential structure Ξ is called a *perennial space* $(\mathbf{X}, \text{Exp}(\mathbf{X}), \Xi)$.

Remark 37. The second condition could be forgotten, as it is redundant with the closure of types under variants. More precisely, it ensures that for every G and H a variant of G , the exponential $!_{[\mu]}^\Xi H$ is equal to $!_{[\nu]}^\Xi H$ for some μ . The addition of this condition allows us to consider only one graphing G in the equivalence class of variants when considering possible

exponentiations. This eases the proofs but does not change the definition of the exponential modality: if Ξ is an “exponential structure” not satisfying this second condition and $\bar{\Xi}$ is the smallest exponential structure containing Ξ (and satisfying this second condition), then $!^\Xi \mathbf{A} = !^{\bar{\Xi}} \mathbf{A}$ (defined below) for all type $\mathbf{A}$.

We now define the exponential connective. The exponential is not defined for all projects, but only on *balanced projects*. We first define this notion, and then introduce some notations useful for the definition of exponential connectives.

Definition 38. A project $\mathfrak{a} = (a, A)$ is *balanced* if $a = 0$, A is slice-free and $D^A = \{0, 1, \dots, k\}$ for an integer k . If E is a set of projects, we write $\text{bal}(E)$ the subset of balanced projects in E .

Notations 39. Let B be a Borel automorphism of $\mathbf{U}$. We denote $B(A)$ the graphing whose edges are $B^{-1} \times \text{Id} \circ \phi \circ B \times \text{Id}$ where the identity is meant on the dialect. When B is a Borel automorphism of $\mathbf{X}$, we abusively denote by $B(A)$ the graphing $\text{Id}_{\mathbf{N}} \times B(A)$.

Definition 40. Let G be a slice-free graphing, with $D^G = \{0, 1, \dots, n\}$ its dialect. Let $\Xi = (((B_p^\mu[n, i])_{0 \leq i \leq n})_{\mu \in I_n})_{n \geq 1})_{P \in \text{Exp}(\mathbf{X})}$ be an exponential structure. We define, for each $P \in \text{Exp}(\mathbf{X})$ and $\mu \in I_n$, the graphing $!_{[\mu]}^\Xi G$ as follows:

- its set of edges is $\{!e \mid e \in E^G\}$;
- each edge $!e$:
 - has source $B^\mu[n, i](S)$ where S is the set such that $S^e = S \times \{i\}$;
 - has target $B^\mu[n, j](T)$ where T is the set such that $T^e = T \times \{j\}$;
 - is realised as $B^\mu[n, j] \circ \psi \circ B^\mu[n, i]^{-1}$, where ψ is the map: $S \rightarrow T$ such that $\phi_e^G(x, i) = (\psi(x), j)$.

Definition 41. Consider a perennial space $(\mathbf{X}, \text{Exp}(\mathbf{X}), \Xi)$. Let $\mathbf{A}$ be a conduct. We define the conduct $!^\Xi \mathbf{A}$ as the set

$$\{!_{[\mu]}^\Xi G \mid G \in \text{bal}(\mathbf{A}), P \in \text{Exp}(\mathbf{X}), \mu \in I_{\text{Card}(D^G)}\}^{\sim \sim}$$

Proposition 42. Consider a perennial space $(\mathbf{X}, \text{Exp}(\mathbf{X}), \Xi)$. For any conduct $\mathbf{A}$, there exists a (successful) project ctr in $!^\Xi \mathbf{A} \otimes !^\Xi \mathbf{A} \multimap !^\Xi \mathbf{A}$

Proof. We refer to our earlier paper on exponentials [Sei14d]. Using the terminology of the latter, let us stress that the conduct $!^\Xi \mathbf{A}$ is a *perennial conduct*, which implies that it can be applied a contraction. $\square$

Remark 43. Every project $\mathfrak{a} = (a, A)$ where $a = 0$ and A is slice-free has a balanced variant. So the exponential of $\mathbf{A}$ is in some ways defined from the set of slice-free projects with null wager in $\mathbf{A}$. This subset of project contains all *successful projects* [Sei14d], i.e. projects that can be interpretations of proofs.

Remark 44. More precisely, if $\text{Card}(D^A) = k$, then it has exactly $k!$ balanced variants. As explained above, the symmetry condition on exponential structures allows us to consider only one of these $k!$ variants to compute the exponential.

This is a very general definition of exponentials. We will now consider a restriction that will allow to prove some generic results in the following. These generic results can be proved upon the hypothesis that the exponential structure under study is *Borel-compatible*, i.e. when two different exponentiations of a given graphing G are equal up to a Borel automorphism of the underlying space.

Definition 45. An exponential structure Ξ is Borel-compatible when for all $\mu, \nu \in I_{\text{Card}(D^G)}$, there exists a Borel automorphism B_μ^ν of $\mathbf{X}$ such that for all balanced graphing G , $!_{[\mu]}^\Xi G = B_\mu^\nu(!_{[\nu]}^\Xi G)$. A perennial space $(\mathbf{X}, \mathbb{E}\text{xp}(\mathbf{X}), \Xi)$ is said Borel-compatible when Ξ is Borel-compatible.

We now consider a property on exponential structures that ensures Borel-compatibility. This definition is in fact incomplete and a bit unsatisfying, so we provide two different properties: weak and strong uniformity. It will be shown that a strongly uniform exponential structure is Borel-compatible. However, it is not clear to the author if the converse holds. In particular, there may be some Borel-compatible non-strongly uniform exponential structures; there may even be such situations where the exponential structure is not weak uniform. However, weak uniformity is a reasonable expectation since it allows for better properties of the model. Indeed, in the case we are considering, i.e. thick graphings with discrete dialects, the dialect actually acts as the discrete measure space with normalised counting measure, i.e. the scalar multiple of the counting measure such that the measure of the whole space equals 1. In this case, weak uniformity implies that the fact that slices were of equal measure is preserved through exponentiation.

Definition 46. An exponential structure is *weakly uniform* if there exists a sequence ζ_n such that for all μ, n, i , $\lambda(X^\mu[n, i]) = \zeta_n$.

It is *strongly uniform* if:

- either for all μ, n, i , $X^\mu[n, i] = X_i^n$;
- or it is weakly uniform and $\mathbf{X}$ is a Polish space.

Proposition 47 (Unitary Equivalence of Exponentials). *A strongly uniform exponential structure is Borel-compatible.*

Proof. By definition of exponential structures, the maps $B^\nu[n, i] \circ B^\mu[n, i]^{-1}$ are partial Borel automorphisms $X^\mu[n, i] \rightarrow X^\nu[n, i]$. We are thus able to construct a partial Borel automorphism $\bigcup_{i=1}^n X^\mu[n, i] \rightarrow \bigcup_{i=1}^n X^\nu[n, i]$ as the disjoint union of these maps. All we need to show now is that this map f extends to a (total) Borel automorphism of $\mathbf{X}$. If for all μ, n, i , $X^\mu[n, i] = X_i^n$ then the obtained map f is already total. If we suppose that there exists real numbers λ_n such that for all μ, n, i , $\mu(X^\mu[n, i]) = \lambda_n$ and that we are working on a Polish space, then the subspaces $\bigcup_{i=1}^n X^\mu[n, i]$ and $\bigcup_{i=1}^n X^\nu[n, i]$ have measure $n\lambda_n$ and their complement is of equal measure. This implies, from the fact that we are working with a Polish space, that there exists a Borel automorphism g from $X - \bigcup_{i=1}^n X^\mu[n, i]$ to $X - \bigcup_{i=1}^n X^\nu[n, i]$, which can be used to extend the map f to the whole space by considering the pairing $f + g$. $\square$

The following definition will be useful in this paper. It defines, for a given exponential structure Ξ , the set of Borel automorphisms that exchange different exponentiations of a given graphing G .

Definition 48. Let Ξ be a Borel-compatible exponential structure. We define the subset $\text{Aut}^\Xi(\mathbf{X})$ as the smallest subset of $\text{Aut}(\mathbf{X})$ – the set $\text{Aut}(\mathbf{X})$ of all Borel automorphisms on $\mathbf{X}$ – such that for all $\mu, \nu \in I_{\text{Card}(D^G)}$, $B_\mu^\nu \in S$.

Now, to define the interpretation of words and machines properly, we will in this paper define exponentials on a space $\mathbf{X}$ and then work on a bigger space $\mathbf{U}$. Even though everything could be done internally, we chose a more structured and (we expect) clearer presentation. The following proposition allows us to lift an exponential structure on $\mathbf{X}$ to an exponential structure on $\mathbf{X} \times \mathbf{Y}$ for any measure space $\mathbf{Y}$. This will allow us to define exponential on a space $\mathbf{X} \times \mathbf{Y}$ by defining exponentials on the simpler space $\mathbf{X}$.

Proposition 49. *If $(\mathbf{X}, \mathbb{E}xp(\mathbf{X}), \Xi)$ is a perennial space, then there exists a canonical structure of perennial space over $\mathbf{X} \times \mathbf{Y}$ for all measure space $\mathbf{Y}$. We write the resulting perennial space $(\mathbf{X} \times \mathbf{Y}, \mathbb{E}xp(\mathbf{X})^\uparrow, \Xi^\uparrow)$ and call it the lifted perennial space. Moreover, if Ξ is Borel-compatible, so is the induced exponential structure $\Xi^\uparrow$.*

Proof. If $\mathbf{X}$ is an exponential space, then for any space $\mathbf{Y}$, the product $\mathbf{X} \times \mathbf{Y}$ can be provided an induced structure of exponential space. Indeed, if P is a perennial situation on $\mathbf{X}$, then considering for all integer n the sets $X_P^n(i) \times \text{Id}_Y$ and the canonical maps $t_P^n[i, j] \times \text{Id}_Y$ defines a perennial situation – the induced perennial situation $P^\uparrow$ – on $\mathbf{X} \times \mathbf{Y}$. We then define $\mathbb{E}xp(\mathbf{X})^\uparrow$ as the set $\{P^\uparrow \mid P \in \mathbb{E}xp(\mathbf{X})\}$.

Similarly, an exponential structure on the exponential space $(\mathbf{X}, \mathbb{E}xp(\mathbf{X}))$ defines an exponential structure on the induced exponential space $(\mathbf{X} \times \mathbf{Y}, \mathbb{E}xp(\mathbf{X})^\uparrow)$. If we write the exponential structure Ξ on $\mathbf{X}$ as $((X_P^\mu[n, i])_{0 \leq i \leq n})_{\mu \in I_n})_{n \geq 1})_{P \in \mathbb{E}xp(\mathbf{X})}$, this induced exponential structure $\Xi^\uparrow$ on $\mathbf{X} \times \mathbf{Y}$ is defined as the families, for each $P^\uparrow \in \mathbb{E}xp(\mathbf{X} \times \mathbf{Y})$:

$$(((B_P^\mu[n, i] \times \text{Id}_Y)_{0 \leq i \leq n})_{\mu \in I_n})_{n \geq 1})$$

Lastly, the preservation of Borel-compatibility is stragithforward, and one can check that $\text{Aut}^{\Xi^\uparrow}(\mathbf{X}) = \{B \times \text{Id}_Y \mid B \in \text{Aut}^\Xi(\mathbf{X})\}$. $\square$

Based on the preceding proposition, we will now fix an perennial space $\mathbf{X}$ and work with the induced perennial space $\mathbf{N} \times \mathbf{X}$, where $\mathbf{N}$ is the set of natural numbers endowed with its counting measure. Finally, we fix the perennial microcosm int generated by all bijections over $\mathbf{N}$ and the smallest perennial microcosm $\text{pet}_\mathbf{X}$ over $\mathbf{X}$. Maps obtained from lifting elements of the microcosm $\text{pet}_\mathbf{X}$ will allow for the definition of an exponential connective, while maps induced by bijections over $\mathbf{N}$ will be used to define word representations. Therefore, the microcosm int is sufficient for the definition of a type $! \text{Words}_\Sigma^{(2)}$ of words over an arbitrary alphabet Σ .

As we will need to represent computation, it will in fact be useful to consider an even larger space than $\mathbf{N} \times \mathbf{X}$. Indeed, the latter contains the necessary structure, encoded in the perennial microcosm int , to represent words. But nothing ensures us that it contains relevant structure for representing interesting computational behaviors. We will therefore consider a measure space $\mathbf{U}$ – the universe – which contains the perennial space $\mathbf{N} \times \mathbf{X}$, i.e. there exists a projection $\pi : \mathbf{U} \rightarrow \mathbf{N} \times \mathbf{X}$. Then each element ϕ of the microcosm int may be extended as a transformation of the whole universe $\mathbf{U}$. Choosing a “coherent” family of extensions of elements of int – i.e. a family which has the structure of a monoid – allows one to define a microcosm over $\mathbf{U}$ which lifts the perennial microcosm int . Conversely, a microcosm $\bar{\text{int}}$ lifts the microcosm int when for each $m \in \text{int}$ there is a unique $\bar{m} \in \bar{\text{int}}$ such that $\pi \circ \bar{m} = m \circ \pi$. We now suppose that such a lifted perennial microcosm $\bar{\text{int}}$ has been chosen once and for all, and we will abusively denote by $\mathbf{X}$ and int the subspace $\iota(\mathbf{N} \times \mathbf{X})$ and the microcosm $\bar{\text{int}}$.

We will also consider that we work in the *macrocosm* $\mathcal{M}$ over the universe $\mathbf{U}$, i.e. the monoid of all bi-mesurable non-singular transformations $\mathbf{U} \rightarrow \mathbf{U}$. Inside this macrocosm sit many microcosms, among them the perennial microcosm int and the smallest perennial microcosm $\text{pet}_\mathbf{X}$ (up to the chosen lifting from $\mathbf{X}$ to $\mathbf{U}$). We will then consider another microcosm $m \subset \mathcal{M}$ which we will use to describe machines. Nothing is asked about the relative positions of $\text{pet}_\mathbf{X}$ and m ; in particular, they may intersect in a non-trivial way.

We now consider, for the rest of this section, that a Borel-compatible perennial space $(\mathbf{X}, \mathbb{E}xp(\mathbf{X}), \Xi)$ has been chosen.

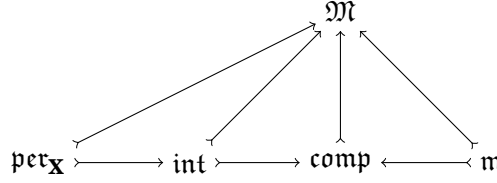


Figure 5: The inclusions of microcosms (hence of models). The macrocosm $\mathfrak{M}$ contains all microcosms, and in particular the microcosm m used to define machines, and the microcosm int used to define integers. The latter contains the smallest perennial microcosm per_X since we actually consider exponentiations of words. The microcosm $comp$ is defined as $m + int$ and is the microcosm in which the computation can be represented.

3.2. Words and Machines. Although an alphabet intuitively contains only a finite number of symbols, the representation of words below can easily be considered for alphabets with infinite (countable) numbers of symbols. Notice that represented computations in this case would still be *finite* as execution is defined as the set of finite (though of unbounded length) paths. The resulting graphings would however be defined on infinite-measure sets as carriers, which may hinder some of the quantitative aspects (this is explained in the author's recent work on exponentials [Sei15a]). Lastly, infinite alphabets implies that machines should be able to have an infinite transition relation, something that would need an adaptation of the forthcoming definition of *predicate machines*.

In this paper, we restrict our attention to a finite alphabets. We now fix such an alphabet Σ until the end of this section. We suppose moreover that the alphabet considered do not contain the symbol $\star$. This condition, which has no effects on the results, will ease the definitions.

Notations 50. We denote by $\Sigma_\star$ the extended alphabet $\Sigma \cup \{\star\}$, by $\Sigma^\dagger$ the set $\Sigma_\star \times \{in, out\}$, and by $\Sigma_{a,r}^\dagger$ the set $\Sigma^\dagger \cup \{a, r\}$, where a (resp. r) stand for accept (resp. reject). We fix once and for all an injection $\Psi : \Sigma_{a,r}^\dagger \rightarrow \mathbf{N}$ and will denote, for $f \in \Sigma_{a,r}^\dagger$ and Y a measurable subset of $\mathbf{X}$, by $\langle f \rangle_Y$ the measurable subset $\Psi(f) \times Y$ of $\mathbf{N} \times \mathbf{X}$. When $Y = X$, we omit the subscript and simply write $\langle f \rangle$. This notation extends to any subset S of $\Sigma_{a,r}^\dagger$: $\langle S \rangle$ is the (disjoint) union of the sets $\langle f \rangle$ for $f \in S$.

The additional symbol $\star$ will be used to represent the starting points of words – or end-markers, and a given word will be represented as a thick graph on the set of vertices $\Sigma^\dagger$. These peculiarities of the interpretation is not an *ad-hoc* choice but comes from the interpretation of normal form lambda-terms in the types of lists.

Take for instance a lambda-term representing the binary word $\star 00101$. This lambda-term is defined as $\lambda f_0. \lambda f_1. \lambda x. f_0(f_0(f_1(f_0(f_1(x)))))$, i.e. it is the term that takes two functions of a given type $X \rightarrow X$ and composes them in order to obtain a new function, say g , of the same type. The symbol $\star$ actually represents the resulting function. The graph representing this binary word can be written from this lambda-term directly. Indeed, we consider a graph with a dialect of cardinality the length of the binary word. We do not give the details of how the dialect appears in the interpretation, but it is due to the interpretation of the contraction rule⁹; a good intuition is that the dialect creates several occurrences of all the functions in

⁹The interested reader will find all the necessary details in previous work [Sei12b, Sei14d].

order to apply them several times – for instance f_0 is applied 3 times in our example. The graph is then obtained by considering that each symbol of the word corresponds to a point in the dialect and:

- writing down the compositions considered in by linking with edges the input and output of the adequate functions f_0, f_1 ;
- linking the input and output of $f_\star$ to the corresponding input and output of the functions f_0 and/or f_1 .

For instance, in our example the cardinality of the dialect is 6 (we count the symbol $\star$) and we obtain the graph shown in Figure 6.

Definition 51. We define the graphs $W_w^{(2)}$ representing a word $w = \star a_1 a_2 \dots a_k \in \Sigma^*$:

$$\begin{aligned} V^{W_w^{(2)}} &= \Sigma^{\uparrow\downarrow} \\ D^{W_w^{(2)}} &= \{0, 1, \dots, k\} \\ E^{W_w^{(2)}} &= \{r, l\} \times \{0, \dots, k\} \\ s^{W_w^{(2)}} &= (r, i) \mapsto (a_i, \text{out}, i) \\ &\quad (l, i) \mapsto (a_i, \text{in}, i) \\ t^{W_w^{(2)}} &= (r, i) \mapsto (a_{i+1}, \text{in}, i+1 \bmod k+1) \\ &\quad (l, i) \mapsto (a_{i-1}, \text{out}, i-1 \bmod k+1) \end{aligned}$$

Later on, we will also consider here a notion of *one-way representation of integers*. These one-way representation do not arise from the proofs-as-programs correspondence as it is the case of the *two-way representations* we explained above. Although they are not obtained as interpretations of proofs since they contain non-invertible edges, these representations of integers are naturally defined in our framework. Anticipating on one-way representations, the interested reader can find in Figure 7 an illustration of the thick graph corresponding to the one-way representation of the word whose two-way representation appears in Figure 6. We refer to subsection 5.3 for a detailed discussion of one-way representations.

Definition 52. Let w be a word $w = \star a_1 a_2 \dots a_k$ over the alphabet Σ . We define the graphing $\bar{W}_w^{(2)}$ of support $\langle \Sigma^{\uparrow\downarrow} \rangle$ and dialect $D^{W_w^{(2)}}$ by the set of edges $E^{W_w^{(2)}}$ and for all edge e :

- the source $S_e^{\bar{W}_w^{(2)}} = \langle X \rangle_f \times \{i\}$ when $s^{W_w^{(2)}}(e) = (f, i)$;
- the target $T_e^{\bar{W}_w^{(2)}} = \langle X \rangle_g \times \{j\}$ when $t^{W_w^{(2)}}(e) = (g, j)$;
- the realiser $\phi_e^{\bar{W}_w^{(2)}} : S_e^{\bar{W}_w^{(2)}} \rightarrow T_e^{\bar{W}_w^{(2)}}, (\Psi(f), x, i) \mapsto (\Psi(g), x, j)$;
- $\omega_e^{\bar{W}_w^{(2)}} = \omega^{W_w^{(2)}}(e)$.

Notations 53. We write $\mathbf{Rep}^{(2)}(w)$ the set of all variants (c.f. Remark 4) of the graphing $\bar{W}_w^{(2)}$, i.e. the set of two-way representations of the word w .

Definition 54. A *two-ways word project* (over Σ) is a couple $(0, L)$ where L is a graphing in $\mathbf{Rep}^{(2)}(w)$. The set of all two-ways word projects over Σ is denoted $\mathbf{W}_\Sigma^{(2)}$.

Definition 55. We define the type (conduct) of words as the set:

$$\mathbf{Words}_\Sigma^{(2)} = (\mathbf{W}_\Sigma^{(2)})^{\perp\perp}$$

We now turn to the notion of machine. We focus in this paper on machines computing predicates, i.e. elements of the type $!\mathbf{Words}_\Sigma^{(2)} \multimap \mathbf{NBool}$. As a machine is traditionally a discrete and finite device, it is quite natural to define a machine as a graph. However, the

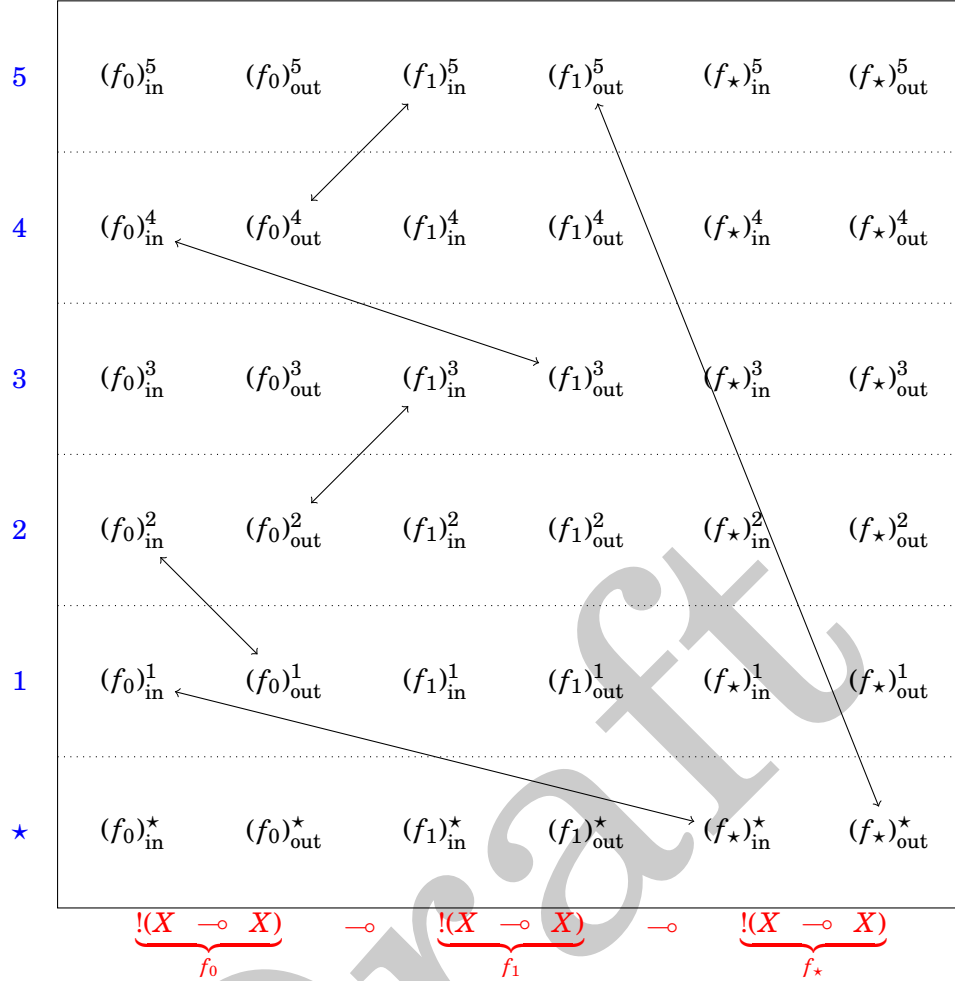


Figure 6: Example: the two-way representation $W_{\star 00101}^{(2)}$ of the binary word $\star 00101$. The dialect is represented on a vertical scale.

machines we consider – called (Ω, m) -machines – will be *realisations* of Ω -weighted graphs as graphings, where the edges are realised by elements of a chosen microcosm m . Intuitively, the notion of *thick graph* underlying this definition corresponds to the simple notion of automaton, while the realisations of edges correspond to particular *instructions*. Although this intuitive understanding of the notion of machines can be followed through the rest of this paper, it should be noted that there are no particular reasons why, in general, a (Ω, m) -machine should be understood as some kind of extension of the notion of automata.

Definition 56. A graphing G is *finite* when there exists a graphing H such that¹⁰ $G \leq H$ and the set of edges E^H is finite.

¹⁰We recall that the notation $G \leq H$ stands for “ F is a refinement of G ” for the notion of refinement informally explained in 2.2. We refer to earlier work by the author for the formal definition [Sei14e].

Definition 57. A *predicate* $(\Omega, \mathfrak{m})$ -machine over the alphabet Σ is a finite Ω -weighted graphing in $\mathfrak{m}$ of support $\langle \Sigma_{a,r}^{\parallel} \rangle$. A *predicate* $(\Omega, \mathfrak{m})$ -machine project is a couple $(0, M)$ where M is a $(\Omega, \mathfrak{m})$ -machine.

In the following, as only predicate machines will be under study, we will simply talk about machines. The term *predicate* $(\Omega, \mathfrak{m})$ -machines was chosen to emphasise that those are meant to compute predicates and that other notions of $(\Omega, \mathfrak{m})$ -machines could be considered.

Remark 58. The deterministic model contains a submodel consisting only of finite graphings. This is a consequence of the fact that the execution of two deterministic finite graphings is finite. In this particular case, we could work in this submodel and forget about the restriction to finite graphings, i.e. a $(\Omega, \mathfrak{m})$ -machine is then just a Ω -weighted graphing in $\mathfrak{m}$ of support $\langle \Sigma_{a,r}^{\parallel} \rangle$ or, conversely, every Ω -weighted graphing in $\mathfrak{m}$ of support $\langle \Sigma_{a,r}^{\parallel} \rangle$ in the model is a $(\Omega, \mathfrak{m})$ -machine. Although quite satisfying, we chose to work with the explicit restriction to finite graphings since the execution of two finite graphings is not finite in general.

3.3. Computation and Tests. In order to state the following proposition, we need to recall some notations on types defined in earlier work. Of particular interest is the type of booleans in the GoI models considered. As discussed in our work on additives [Sei14b], the right type for representing booleans in our framework is to consider $\mathbf{Bool} = \mathbf{T}_{\text{true}} \oplus \mathbf{T}_{\text{false}}$. This type is defined up to an arbitrary choice of (disjoint) sets (here measurable subsets) *true* and *false*; in the following, we will naturally consider the sets $\langle a \rangle$ and $\langle r \rangle$ to play the role of *true* and *false*. Moreover, the type $\mathbf{T}_X$ where X is a (measurable) set, is the type of all projects of carrier X . The following proposition shows that a $(\Omega, \mathfrak{m})$ -machine is an element of the type $!\mathbf{Words}_{\Sigma}^{(2)} \multimap \mathbf{NBool}$, where $\mathbf{NBool} = \mathbf{T}_{\langle a,r \rangle} = \mathbf{T}_{\langle a \rangle} \& \mathbf{T}_{\langle r \rangle}$ is a type we consider as a kind of *non-deterministic booleans*.

Proposition 59. Every $(\Omega, \mathfrak{m})$ -machine $(0, M)$ is an element of the type $!\mathbf{Words}_{\Sigma}^{(2)} \multimap \mathbf{NBool}$.

Proof. This is more a remark than a real proposition. Since $\mathbf{NBool}$ contains all projects over the carrier $\langle a, r \rangle$, it is clear that it contains the result of the computation of a $(\Omega, \mathfrak{m})$ -machine on a word representation: one only has to check that by definition the resulting project is of carrier $\langle a, r \rangle$. $\square$

Definition 60 (Computation). Let $\mathfrak{M} = (0, M)$ be a $(\Omega, \mathfrak{m})$ -machine project over the alphabet Σ , $w \in \Sigma^*$ and $\{w\} = (0, \mathbb{I}_{[\mu]}^{\Xi} \bar{W}_w^{(2)}) \in !\mathbf{Words}_{\Sigma}^{(2)}$. The *computation* of $\mathfrak{M}$ over $\{w\}$ is defined as the project $\mathfrak{M} :: \{w\}$, which is a graphing over $\langle a, r \rangle$.

We now introduce the notion of *test*. This notion is essential as it allows for considering several notions of acceptance. Even though acceptance may be defined “by hand” by describing directly the expected result, the definition through tests allows for a more interesting definition. Indeed, the acceptance is described inside the model, using already existing notions, i.e. we do not modify the models to define testing. In other words, acceptance and rejection are given a logical meaning, as testing is tied with the process of constructing types.

Definition 61 (Tests). A *test* is a family $\mathcal{T} = \{(t_i, T_i) \mid i \in I\}$ of projects of support $\langle a, r \rangle$ satisfying the following uniformity condition: for all $(t_i, T_i) \in \mathcal{T}$ and for all B in $\text{Aut}^{\Xi}(\mathbf{X})$, the project $(t_i, B(T_i))$ is also in $\mathcal{T}$.

Remark 62. This notion of uniformity is essential. If tests were not uniform, one may be able to distinguish between two different representations of a given word.

Let us discuss this notion of test. A test $\mathcal{T} = \{T_i\}_{i \in I}$ defines a subtype of **NBool** defined as $\mathbf{Pass}_{\mathcal{T}} = \{T_i \mid i \in I\}^\perp$. A machine M accepts the word w if and only if $M :: !\bar{W}_w^{(2)} \perp T_i$ for all $i \in I$, i.e. $M :: !\bar{W}_w^{(2)} \in \mathbf{Pass}_{\mathcal{T}}$. In particular, the set $\{M :: T \mid T \in \mathcal{T}\}$ can be used to define a subtype of $!\mathbf{Words}_{\Sigma}^{(2)}$, namely $\mathbf{Lang}_{\mathcal{T}}^M$ representing the language of M , i.e. the subtype of $!\mathbf{Words}_{\Sigma}^{(2)}$ generated by all words w such that $M :: !\bar{W}_w^{(2)} \in \mathbf{Pass}_{\mathcal{T}}$.

This notion of test is quite versatile. We illustrate this by considering some simple tests as examples; to simplify this discussion we consider that the machines considered all have an edge from both accept and reject to their initial state, which will be a peculiarity of the encoding of machines. A simple example is the test defined as the thick graph with a single loop on the vertex accept with m defined as $m(1) = 1$. To be orthogonal to this test, $T :: W_w^{(2)}$ must possess at least an edge from accept to itself; this test can be used to represent the acceptance for deterministic classes and for non-deterministic ones such as NLOGSPACE. With the same measurement m , the test defined as a graph with a single loop on reject together with the wager -1 will represent non-deterministic classes such as CONLOGSPACE: being orthogonal to this test means that there are no edges from reject to itself.

This notion of test will moreover adapt to other computational paradigms. In particular, working with the measurement $m(x) = -\log(1-x)$ studied in earlier constructions [Sei12a], one can define the test defined by the family of projects whose graphs have a single loop on the accept vertex and wagers are all real numbers between 0 and $-\log(2)$. One can then check that being orthogonal to all these projects amounts to having a number of edges from accept to accept in $M :: W_w^{(2)}$ whose added weights is greater than¹¹ $\frac{1}{2}$. Hence, this family of tests can be used to characterise unbounded probabilistic complexity classes such as PLOGSPACE or PPTIME. This will turn out to be exact as we will show later on how to characterise the class PLOGSPACE by using this exact test.

We now want to define the language characterised by a machine. For this, one could consider *existential* $\mathcal{L}_{\exists}^{\mathcal{T}}(M)$ and *universal* $\mathcal{L}_{\forall}^{\mathcal{T}}(M)$ languages for a machine M w.r.t. a (non-necessarily uniform) test $\mathcal{T}$:

$$\begin{aligned} \mathcal{L}_{\exists}^{\mathcal{T}}(M) &= \{w \in \Sigma^* \mid \forall T_i \in \mathcal{T}, \exists W_w \in \mathbf{Rep}^{(\cdot)}(w), M :: W_w \perp T_i\} \\ \mathcal{L}_{\forall}^{\mathcal{T}}(M) &= \{w \in \Sigma^* \mid \forall T_i \in \mathcal{T}, \forall W_w \in \mathbf{Rep}^{(\cdot)}(w), M :: W_w \perp T_i\} \end{aligned}$$

We now introduce the notion of uniformity, which describes a situation where both definitions above coincide. This collapse of definitions is of great interest because it ensures that both of the following problems are easy to solve:

- whether a word belongs to the language: from the existential definition one only needs to consider one representation of the word;
- whether a word does not belong to the language: from the universal definition, one needs to consider only one representation of the word.

Definition 63 (Uniformity). Let m be a microcosm, Ω a monoid and Ξ an exponential structure. The test $\mathcal{T}$ is said *uniform* w.r.t. (Ω, m) -machines if for all such machine M , and any two elements $!_{[\mu]}^{\Xi} W, !_{[\nu]}^{\Xi} W'$ for $W, W' \in \mathbf{Rep}^{(\cdot)}(w)$, one has:

$$(0, M) :: (0, !_{[\mu]}^{\Xi} W) \perp \mathcal{T} \text{ if and only if } (0, M) :: (0, !_{[\nu]}^{\Xi} W') \perp \mathcal{T}$$

¹¹We recall [Sei12a] that the measurement $-\log(1-x)$ satisfies the property that $G \perp H$ if and only if $\angle(G) \perp H$, where $\angle(G)$ is the graph G where the (non-empty) sets of edges $\{f_i\}_{i \in J}$ from a vertex v to a vertex v' was replaced by a single edge e from v to v' whose weight $\omega(e) = \sum_{i \in J} \omega(f_i)$.

Now, suppose given a notion of machine and a uniform test w.r.t. this notion of machine. One can define the language $\mathcal{L}^{\mathcal{T}}(M)$ accepted by a machine M by testing the result of the execution of M with representations of words. The choice of the representations of words does not change the result of this test since we assumed uniformity, and the following definition thus makes sense.

Proposition 64. *Let $\mathfrak{m}$ be a microcosm, Ω a monoid and Ξ an exponential structure. If the test $\mathcal{T}$ is uniform w.r.t. $(\Omega, \mathfrak{m})$ -machines, $\mathcal{L}_{\Xi}^{\mathcal{T}}(M) = \mathcal{L}_{\mathbb{V}}^{\mathcal{T}}(M)$. We thus use the notation $\mathcal{L}^{\mathcal{T}}(M)$. If $\mathbb{M}$ is a set of $(\Omega, \mathfrak{m})$ -machines, we set $\mathbb{L}^{\mathcal{T}}(\mathbb{M}) = \{\mathcal{L}^{\mathcal{T}}(M) \mid M \in \mathbb{M}\}$.*

Moreover, uniformity implies that a machine M will accept or reject an input on the sole base of the word it is representing, i.e. its answer does not depend on the choice of the representation. In other words, requesting uniformity is equivalent to asking that all machines M implement a *function* from words to booleans, regardless of the fact that many different representations of a given word can be given as input.

Showing that a test is uniform is in general a quite complicated thing, as it involves the computation over all representations of words. We now describe a general situation which allows to prove uniformity of tests, in the case of uniform exponentials. I.e. we consider a property of spaces w.r.t microcosms which ensures the uniformity of all tests. This will be used in the next section to show the uniformity of the considered tests.

If it is supposed that tests are graphings over a sub-microcosm of $\mathfrak{m}$, then this property can be ensured by a property depending only on the inclusion $\iota : \mathbf{X} \rightarrow \mathbf{U}$ and the microcosm $\mathfrak{m}$.

Definition 65. A projection $\pi : \mathbf{U} \rightarrow \mathbf{X}$ has the *lifting property w.r.t. the microcosm $\mathfrak{m}$* if for all bimeasurable bijective transformation $\theta : \mathbf{X} \rightarrow \mathbf{X}$, there exists a bimeasurable bijective transformation $\Theta : \mathbf{U} \rightarrow \mathbf{U}$ such that $\theta \circ \pi = \pi \circ \Theta$ and for all $\phi \in \mathfrak{m}$, $\phi \circ \Theta = \Theta \circ \phi$.

The following lemma is immediate from the definition of the lifting property.

Lemma 66 (Downward closure of the lifting property). *Suppose that $\pi : \mathbf{U} \rightarrow \mathbf{X}$ has the lifting property w.r.t. the microcosm $\mathfrak{m}$, and that $\mathfrak{n} \subset \mathfrak{m}$. Then $\pi : \mathbf{U} \rightarrow \mathbf{X}$ has the lifting property w.r.t. the microcosm $\mathfrak{n}$.*

The lifting property can be defined for sets of measurable maps instead of microcosms since the monoid structure is not used in the definition. This leads to the following lemma which will help prove that a microcosm has the lifting property.

Lemma 67 (Generating sets and the lifting property). *Suppose that $\pi : \mathbf{U} \rightarrow \mathbf{X}$ has the lifting property w.r.t. a generating subset $m \subset \mathfrak{m}$. Then $\pi : \mathbf{U} \rightarrow \mathbf{X}$ has the lifting property w.r.t. the microcosm $\mathfrak{m}$.*

Proof. By definition, for all bimeasurable bijective transformation $\theta : \mathbf{X} \rightarrow \mathbf{X}$, there exists a bimeasurable bijective transformation $\Theta : \mathbf{U} \rightarrow \mathbf{U}$ such that for all $\phi \in m$, $\phi \circ \Theta = \Theta \circ \phi$ and $\pi \circ \Theta = \theta \circ \pi$. Now, if φ is an element of $\mathfrak{m}$, then it is a composite $\varphi = \phi_1 \circ \dots \circ \phi_k$ of elements of m . By induction, one can then show that $\varphi \circ \Theta = \phi_1 \circ \dots \circ \phi_k \circ \Theta = \Theta \circ \phi_1 \circ \dots \circ \phi_k = \Theta \circ \varphi$. $\square$

Proposition 68. *Let Ω be a Borel-compatible exponential structure. If the projection $\pi : \mathbf{U} \rightarrow \mathbf{X}$ has the lifting property w.r.t. the microcosm $\mathfrak{m}$, then any test is uniform w.r.t. $(\Omega, \mathfrak{m})$ -machines.*

Proof. We fix a $(\Omega, \mathfrak{m})$ -machine M' and a test $\mathcal{T}$. We want to show that $(0, M) :: !_{[\mu]}^{\Xi} \{\mathfrak{w}\} \downarrow \mathcal{T}$ if and only if $(0, M) :: !_{[\nu]}^{\Xi} \{\mathfrak{w}\} \downarrow \mathcal{T}$. We will consider the computations (c, C) and (c', C') of $(0, M)$ over $!_{[\mu]}^{\Xi} \{\mathfrak{w}\}$ and $!_{[\nu]}^{\Xi} \{\mathfrak{w}\}$ respectively; notice that C and C' are graphings in $\mathfrak{m}$. Since

orthogonality is based upon cycles, the wanted property holds as long as there exists a bijective correspondence between alternating cycles between C and an element $\mathfrak{T}_i = (t_i, T_i) \in \mathcal{T}$ and alternating cycles between C' and another element $\mathfrak{T}_j = (t_j, T_j) \in \mathcal{T}$. Now, from Borel-compatibility, we have that $!_{[\mu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$ is obtained by $!_{[\nu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$ through a Borel automorphism $\theta : \mathbf{X} \rightarrow \mathbf{X}$. By assumption, θ can be lifted to a Borel automorphism Θ over the universe $\mathbf{U}$ in such a way that for all $\phi \in \mathfrak{m}$, $\phi \circ \Theta = \Theta \circ \phi$. Then an edge in C' is of the form

$$\phi_{e_1}^M \circ \Theta \circ \phi_{f_1}^{!_{[\nu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}} \circ \Theta^{-1} \circ \phi_{e_2}^M \circ \dots \circ \Theta \circ \phi_{f_k}^{!_{[\nu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}} \circ \Theta^{-1}$$

which can be rewritten as

$$\Theta \circ \phi_{e_1}^M \circ \phi_{f_1}^{!_{[\nu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}} \circ \Theta^{-1} \circ \Theta \circ \phi_{e_2}^M \circ \dots \circ \phi_{f_k}^{!_{[\nu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}} \circ \Theta^{-1}$$

By simplifying $\Theta^{-1} \circ \Theta$, we obtain $\Theta \circ \phi_{e_1}^M \circ \phi_{f_1}^{!_{[\nu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}} \circ \phi_{e_2}^M \circ \dots \circ \phi_{f_k}^{!_{[\nu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}} \circ \Theta^{-1}$, i.e. it is the image of an edge in C through the Borel automorphism B . Now, if π is an alternating cycle between C' and T_j , then through the rewriting of the edges just considered, it can be rewritten as an alternating cycle between C and $\Theta(T_j)$. It is not difficult to show that this correspondence between cycles is bijective, showing that $(c, C') \perp \mathfrak{T}_j$ if and only if¹² $(c, C) \perp \Theta(\mathfrak{T}_j)$. Since $\Theta(\mathfrak{T}_j) \in \mathcal{T}$ if and only if $\mathfrak{T}_j \in \mathcal{T}$, the proof is finished. $\square$

4. FROM REGULAR to LOGSPACE

4.1. The models. We now consider the measure space $\mathbf{U} = \mathbf{N} \times [0, 1]^{\mathbf{N}}$ with its Borel σ -algebra and the Lebesgue measure. To define microcosms, we use the constructor $+$: recall that if $\mathfrak{m}$ and $\mathfrak{n}$ are two microcosms, $\mathfrak{m} + \mathfrak{n}$ is the smallest microcosm containing both $\mathfrak{m}$ and $\mathfrak{n}$. Moreover, we will need to explain which map $m : \Omega \rightarrow \mathbf{R}_{\geq 0} \cup \{\infty\}$ we are choosing for defining orthogonality. All results will be stated by specifying a property on m , rather than fixing a specific value for it in order to be as general as possible. However, all the considered properties are shared by the map $m : x \mapsto -\log(1 - x)$ which defines an orthogonality which was shown [Sei12a] to relate to the so-called *Fuglede-Kadison determinant* [FK52].

4.1.1. Exponential. We consider the perennial situation on $[0, 1]$ with its Borel σ -algebra and Lebesgue measure which is described in Remark 33. Each measure-preserving Borel automorphism $B \in \text{Aut}([0, 1], \mu)$ defines a new perennial structure P_B by defining $X_{P_B}^n(i) = B(X_P^n(i))$ and $t_{P_B}^n[i, j] = B(t_P^n[i, j])$. We define the exponential space $[0, 1]$ with the family $\text{Exp}([0, 1]) = \{P_B \mid B \in \text{Aut}([0, 1], \mu)\}$. The smallest perennial microcosm is then the set of all measure-preserving Borel automorphisms. The exponential structure Ξ is then defined by the family of Borel automorphisms $E_{P_B}[n, i] : [0, 1] \rightarrow B([i/(n+1), (i+1)/(n+1)], x \mapsto B((x+i)/(n+1)))$. This exponential structure is then strongly uniform, and therefore Borel-compatible.

As explained above, this induces a lifted perennial space $(\mathbf{N} \times [0, 1], \text{Exp}([0, 1])^\dagger, \Xi^\dagger)$. This perennial space will be used for word representations. To consider interesting notions of machines, we will work in the universe $\mathbf{U} = \mathbf{N} \times [0, 1]^{\mathbf{N}}$, as explained above, where the perennial space $\mathbf{N} \times [0, 1]$ is the image of the projection $(a, (s_0, s_1, \dots)) \mapsto (a, s_0)$. The lifted microcosm

¹²Remark the preceding rewriting of paths shows that alternating cycles between M and $!_{[\nu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$ and alternating cycles between M and $!_{[\mu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$ are equal, hence $c = c'$.

int we consider is simply the microcosm of all maps $s \times \text{Id}_{[0,1]^{\mathbf{N}}}$ for $s \in \text{int}$, i.e. the set of all $s \times \text{Id}_{[0,1]^{\mathbf{N}}}$ for s a measure-preserving Borel automorphism on $\mathbf{N} \times [0, 1]$.

4.1.2. Machines. We now define the following microcosms which will be used to describe machines:

- $\mathfrak{m}_1$ is the monoid induced by bijections on $\mathbf{N}$: $\tau_f : (n, x) \mapsto (f(n), x)$;
- $\mathfrak{m}_{i+1}$ is the monoid $\mathfrak{m}_i + \mathfrak{s}_{i+1}$ where $\mathfrak{s}_{i+1}$ is the monoid generated by the single map:

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

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

First, we show that the projection $\pi : \mathbf{N} \times [0, 1]^{\mathbf{N}} \rightarrow \mathbf{N} \times [0, 1]$ has the lifting property w.r.t. the microcosm $\mathfrak{m}_\infty$ (Definition 65). This will imply that any submicrocosm has the lifting property as well by Lemma 66.

Proposition 69. *For all bimeasurable bijective transformation $\theta : \mathbf{N} \times [0, 1] \rightarrow \mathbf{N} \times [0, 1]$, there exists a bimeasurable bijective transformation $\Theta : \mathbf{N} \times [0, 1]^{\mathbf{N}} \rightarrow \mathbf{N} \times [0, 1]^{\mathbf{N}}$ such that for all $\phi \in \mathfrak{m}_\infty$, $\phi \circ \Theta = \Theta \circ \phi$ and $\pi \circ \Theta = \theta \circ \pi$.*

Proof. We pick such a bimeasurable bijective transformation θ . Notice that since $\mathbf{N}$ is discrete, this amounts to pick a bijection on $\mathbf{N}$, say ϕ , and a bimeasurable bijective transformation on $[0, 1]$, say θ_0 . I.e., $\theta = \phi \times \theta_0$. We then define the following bimeasurable bijective transformation Θ :

$$\Theta : \begin{cases} \mathbf{N} \times [0, 1]^{\mathbf{N}} & \rightarrow & \mathbf{N} \times [0, 1]^{\mathbf{N}} \\ (n, (s_i)) & \mapsto & (\phi(n), (\theta_0(s_i))) \end{cases}$$

It is clear that $\pi \circ \Theta = \theta \circ \pi$. Moreover, using Lemma 67, the fact that $\phi \circ \Theta = \Theta \circ \phi$ is a simple verification: we compute both sides of the equation with the map $\phi = s_i$ for an integer $i \geq 2$ and notice they are equal. $\square$

4.2. Two-Way Multihead Automata. In the following, we consider a variant of the classical notion of two-way multihead finite automata obtained by:

- fixing once and for all the right and left end-markers as being equal to the fixed symbol $\star$;
- fixing once and for all unique initial, accept and reject states, respectively denoted by init , accept and reject ;
- choosing that each transition step involves the movement of at most one of the multiple heads of the automaton.

It should be clear that these choices in the design of the automata have no effect on the sets of languages recognized.

Definition 70. A two-way multihead automaton M with k heads is defined as a tuple $(\Sigma, Q, \rightarrow)$, where $\rightarrow \subseteq (\Sigma_\star^k \times Q) \times ((\{1, \dots, k\} \times \{\text{in}, \text{out}\}) \cup \{\epsilon\}) \times Q$ is the *transition relation* of M . The automaton M is *deterministic* when the relation $\rightarrow$ is functional.

The set of two-way multihead automata with k heads is written $\mathbf{2nfa}(k)$, and the set of all two-way multihead automata $\cup_{k \geq 1} \mathbf{2nfa}(k)$ is denoted by $\mathbf{2nfa}$. The set of deterministic two-way multihead automata with k heads is written $\mathbf{2dfa}(k)$, and the set of all deterministic two-way multihead automata $\cup_{k \geq 1} \mathbf{2dfa}(k)$ is denoted by $\mathbf{2dfa}$.

Definition 71. We denote, for all integer k , by $2\text{NFA}(k)$ (resp. $2\text{DFA}(k)$) the set of languages accepted by multihead automata in $\mathbf{2nfa}(k)$ (resp. in $\mathbf{2dfa}(k)$).

Remark 72. Notice that the sets $2\text{NFA}(1)$ and $2\text{DFA}(1)$ were shown equal by Rabin and Scott [RS59]. The set of languages $\text{REGULAR} = 2\text{NFA}(1) = 2\text{DFA}(1)$ is usually called the set of *regular languages*.

The following two results are most important in the theory of two-way multihead automata.

Theorem 73 (Monien [Mon76]). *For all k , the set $2\text{NFA}(k)$ (resp. $2\text{DFA}(k)$) is a strict subset of $2\text{NFA}(k+1)$ (resp. $2\text{DFA}(k+1)$).*

Theorem 74. *The set of languages recognized by all multihead automata (resp. all deterministic multihead automata) is equal to NLOGSPACE (resp. LOGSPACE):*

$$\bigcup_{k \geq 1} 2\text{NFA}(k) = \text{NLOGSPACE} \qquad \bigcup_{k \geq 1} 2\text{DFA}(k) = \text{LOGSPACE}$$

4.3. Acyclicity. One noticeable gap between the general notion of multihead automata and their realisation is that of typing. If an automaton – or rather the realisation of this automaton – can be typed, then it must be *acyclic*. This notion of acyclicity is extremely strong, but corresponds to logical constraints that ensures in some way that the machine can be typed, i.e. that one can ensure its termination. Indeed, recall that any $(\Omega, \mathfrak{m})$ -machine is an element of the type $\mathbf{!Words}_{\Sigma}^{(2)} \multimap \mathbf{NBool}$; this is because the type $\mathbf{NBool}$ is extremely permissive, as it is a sort of “universal type” since any project whose carrier is a subset of $\text{accept} \cup \text{reject}$. We will therefore now focus on a restricted set of machines, that is the machines that can be typed $\mathbf{!Words}_{\Sigma}^{(2)} \multimap \mathbf{Bool}$ (this type will change when considering non-deterministic and probabilistic machines later on). For technical (more precisely logical) reasons, the machines that are elements of a type such as $\mathbf{Words}_{\Sigma}^{(2)} \multimap \mathbf{Bool}$ are necessarily *acyclic* – the definition is found below, i.e. no cycles appear between M and $\bar{W}_{\mathfrak{w}}^{(2)}$ for any word $\mathfrak{w}$.

Definition 75. A $(\Omega, \mathfrak{m})$ -machine M is *acyclic* over Σ when for all word $\mathfrak{w} \in \Sigma^*$ and all $\mu \in I_{|\mathfrak{w}|}$, $\llbracket M, \bar{!}_{[\mu]}^{(2)} \bar{W}_{\mathfrak{w}}^{(2)} \rrbracket_m = 0$. By extension, any graphing G satisfying this property is called *acyclic*.

Acyclicity could seem a strong hypothesis. However, it is necessary as it comes from the fact that all we are doing is logic-related. As a consequence, all machines we are representing can be typed, i.e. provably compute binary predicates. The fact that it is provable implies that the machine satisfies some constraint, namely acyclicity, allowing it to be typed. The notion of acyclicity is quite natural in this perspective. We will introduce later on the corresponding notion (also named “acyclicity”) on automata.

Proposition 76. *A project (a, A) in $\mathbf{Bool}$ satisfies that $a = 0$.*

Proof. The conduct $\mathbf{Bool}$ is a proper behavior. Hence any $(a, A) \in \mathbf{Bool}$ is such that $a = 0$. $\square$

Proposition 77. *A project (m, M) in $\mathbf{!Words}_{\Sigma}^{(2)} \multimap \mathbf{Bool}$ is such that $m = 0$ and M is acyclic.*

Proof. First, $\mathbf{!Words}_{\Sigma}^{(2)} \multimap \mathbf{Bool}$ is a proper behavior, as the $\mathfrak{A}$ of a coperennial conduct and a behavior. Hence any $(m, M) \in \mathbf{Bool}$ is such that $m = 0$. Secondly, $\llbracket M, \bar{!}_{[\mu]}^{(2)} \bar{W}_{\mathfrak{w}}^{(2)} \rrbracket_m = 0$ because the wager of $M :: \bar{!}_{[\mu]}^{(2)} \bar{W}_{\mathfrak{w}}^{(2)}$ is equal to $\llbracket M, \bar{!}_{[\mu]}^{(2)} \bar{W}_{\mathfrak{w}}^{(2)} \rrbracket_m$ (wagers are equal to zero) and should be equal to 0 from the preceding proposition. $\square$

Remark 78. Following Remark 58, if we were to work in the finite deterministic model, the last proposition would imply that all elements of the type $!\mathbf{Words}_{\Sigma}^{(2)} \multimap \mathbf{Bool}$ are acyclic (Ω, m) -machines.

This is why we introduce a corresponding notion of *acyclicity* for automata.

Definition 79. An automaton M is *cyclic against a w* when there exists a configuration s (i.e. a choice of heads positions on the input and a state) and a sequence of transitions $t_0, \dots, t_n$ such that

$$s \xrightarrow{t_0} s_1 \xrightarrow{t_1} \dots \xrightarrow{t_{n-1}} s_n \xrightarrow{t_n} s$$

An automaton M is *acyclic against w* if it is not cyclic against w . An automaton is *acyclic* if it is acyclic against any input w .

Example 80. This is a very strong notion of acyclicity since we are excluding all possible cycles, even if they will never be reached. For instance, consider a simple k -head automaton – acyclic or not – for recognizing your favorite language $\{\star a_1 \dots a_n \mid \text{Card}(\{i \in \{1, \dots, n\} \mid a_i = 0\})\}$. Now, extend it with a new state ϵ and a transition $(\vec{a}, \epsilon) \rightarrow (\emptyset, \epsilon)$, where $\vec{a} \in \{\star, 0, 1\}^k$ and $\emptyset$ represent the fact that no head is moving. This new automaton is cyclic even though no execution trace will go through the cycle created by ϵ .

Remark 81. If we consider the acyclic versions of **2dfa**, then the separation result stated in Theorem 73 is weakened. Indeed, if we write becomes $2\text{ADFA}(k)$ the set of languages accepted by *acyclic* elements of **2dfa**(k), then we obtain a strict inclusion of $2\text{ADFA}(k)$ into $2\text{ADFA}(2k+2)$ and not in $2\text{ADFA}(k+1)$ since we need to add pointer to enforce acyclicity. The author believes, however, that the strict inclusion $2\text{ADFA}(k) \subsetneq 2\text{ADFA}(k+1)$ holds.

4.4. Realising acyclic 2nfa. There are two main differences between the model of multi-head automata with i heads and the notion of (Ω, m_i) -machines.

- The first difference comes from the fact that the computation of (Ω, m_i) -machines is “dynamic”, i.e. corresponds to a dialogue between the machine and the representation of the word it is given as input. As a consequence, one has the knowledge of what symbol a given pointer is reading at a given location *only at the exact moment the pointer moves onto this location*. I.e. the pointer receives information about the input from the integer, and one has to store it if it is to be reused later on. This is different from the way multihead automata work since the latter can, at any given time, access the value located where any head is pointing at. This difference can be taken care of by considering an extended set of states. As a consequence, the automaton M with a set of states Q will be realised as a (Ω, m_i) -machine with an extended set of states $\bar{Q} \times \{\star, 0, 1\}^i$.
- The second difference is that when one “moves the i -th head” of a (Ω, m_k) -machine, it induces a reindexing of the sets of heads. I.e. a (Ω, m_k) -machine should be understood as a multihead automata that can only move its principal head, but has the possibility of reindexing its heads following any permutation over k elements. To deal with this, we could consider (yet another) variation on multihead automata to make them mimic (Ω, m_k) -machines and then show they are equivalent to the usual notion of multihead automata (which they are). We chose however to interpret directly the usual notion of multihead automata as (Ω, m_k) -machines, and conversely, dealing with this issue in the proofs. As a consequence, the extended set of states $\bar{Q}$ is obtained as $Q \times \mathfrak{S}_k$.

Now, let M be a deterministic automaton with k heads. We here write $M = (\Sigma, Q, \rightarrow)$. We will define $\{M\}$ a graphing in $\mathfrak{m}_k$ with dialect – set of states – $Q \times \mathfrak{G}_k \times \{\star, 0, 1\}^k$ as follows. The set of edges of $\{M\}$ is the set:

$$E^{\{M\}} = \{(\{t\}, a, d, \sigma) \mid t \in \rightarrow, a \in \{\star, 0, 1\}, d \in \{\text{in}, \text{out}\}, \sigma \in \mathfrak{G}_k\}$$

The source of the edge $(\{t\}, a, d, \sigma)$ for $t = ((\vec{s}, q), (i, d', q'))$ is:

- the set $(\langle\langle s, d \rangle\rangle, q, \sigma, \vec{s})$ when $q \neq \text{init}$;
- the set $(\langle a \rangle, \text{init}, \text{Id}, \vec{\star})$ if $q = \text{init}$ and $d = \text{in}$;
- the set $(\langle r \rangle, \text{init}, \text{Id}, \vec{\star})$ if $q = \text{init}$ and $d = \text{out}$.

The target of the edge $(\{t\}, a, d, \sigma)$ for $t = ((\vec{s}, q), (i, d', q'))$ is:

- the set $(\langle\langle s_i, d' \rangle\rangle, q', \tau_{1,i} \circ \sigma, \vec{s}[s_{\sigma^{-1}(1)} := a])$ when $q' \notin \{\text{accept}, \text{reject}\}$;
- the set $(\langle q' \rangle, \text{init}, \text{Id}, \vec{\star})$ if $q' \in \{\text{accept}, \text{reject}\}$.

The realiser of the edge $(\{t\}, a, d, \sigma)$ for $t = ((\vec{s}, q), (i, d', q'))$ is the map s_i composed with the adequate bijection on $\mathbf{N}$. E.g. when $q \neq \text{init}$ and $q' \notin \{\text{accept}, \text{reject}\}$ it is the map s_i composed with the bijection exchanging $\langle\langle s, d \rangle\rangle$ and $\langle\langle s_i, d' \rangle\rangle$.

Proposition 82. *The alternating paths of odd length between $\{M\}$ and a exponentiated representation $!_{[\mu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$ of a word $\mathfrak{w}$ of source $\langle a \rangle$ or $\langle r \rangle$ are in bijective correspondence with the non-empty computation traces of M given $\mathfrak{w}$ as input.*

Proof. Suppose there is a computation trace $\mathcal{T}$ of M on input $\mathfrak{w} = \star \alpha_1 \dots \alpha_n$ that reaches a configuration $(i_1, \dots, i_k, q)$ where $i_1, \dots, i_k$ are positions of the k heads of M on the input and q is a state. We write $\vec{\alpha}$ the sequence $\alpha_{i_1}, \alpha_{i_2}, \dots, \alpha_{i_k}$. We will show that there is an alternating path of odd length from $\langle a \rangle$ (resp. $\langle r \rangle$) to $(\langle\langle \alpha_i, d \rangle\rangle, q, \sigma, \vec{\alpha})$ where σ is the permutation of heads applied since the start of the computation if and only if there is a transition $t = ((\vec{\alpha}, q), (i, d, q'))$ in the automaton M .

This is shown by induction. For the empty trace, the corresponding paths are the length 1 paths starting at $\langle a \rangle$ and $\langle r \rangle$ respectively.

Now, suppose the hypothesis is true for a computational trace as described above, and write π the corresponding alternating path. We will show that any extension of this computational trace corresponds to an extension of the path π and conversely. The transition $((\vec{\alpha}, q), (i, d, q'))$ leads to the configuration $(i_1, \dots, i_{i-1}, i_i + \epsilon(d), i_{i+1}, \dots, i_k, q')$, where $\epsilon(\text{in}) = 1$ and $\epsilon(\text{out}) = -1$. We write a the symbol $\alpha_{i_i + \epsilon(d)}$.

The path π can be extended by the edge e of the word representation of source $B^\mu[n, i](\langle\langle \alpha_i, d \rangle\rangle)$. This edge is of target $B^\mu[n, i + \epsilon d](\langle\langle a, \vec{d} \rangle\rangle)$, and the path πe is of target $T = (\langle\langle a, \vec{d} \rangle\rangle, q, \sigma, \vec{\alpha})$. There is an edge of source T in $\{M\}$ if and only if there exists a transition $t = ((\vec{\alpha}[a_i = a], q), (j, d', q'))$, namely the transition $(\{t\}, a, \vec{d}, \sigma)$. In that case, this transition has target $(\langle\langle \alpha_j, d' \rangle\rangle, q', \tau_{1,j} \circ \sigma, \vec{\alpha}[a_i = a])$.

What we just shown is enough to show the existence of a bijective correspondence between traces $\mathcal{T} \cdot t$ (note that $\mathcal{T}$ can be empty) and odd length alternating paths starting at $\langle a \rangle$ (resp. $\langle r \rangle$). $\square$

The following corollary is then quite easy to obtain.

Corollary 83. *The automaton M accept the word $\mathfrak{w}$ if and only if there exists a path from $\langle \text{accept} \rangle$ to itself.*

Remark 84. Acyclicity is not needed here, as this result is independent from typing!

Proposition 85. *The automaton M is acyclic if and only if $\{M\}$ is acyclic.*

Proof. This result is shown using a similar technique as Proposition 82. We write $\mathfrak{w} = \star \alpha_1 \dots \alpha_n$, consider a configuration $(i_1, \dots, i_k, q)$ and write $\vec{\alpha}$ the sequence $\alpha_{i_1}, \alpha_{i_2}, \dots, \alpha_{i_k}$. We are interested in alternating cycles between $\{M\}$ and $!_{[\mu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$.

Notice that one can adapt the proof of Proposition 82 to show that computation traces starting in configuration $(i_1, \dots, i_k, q)$ are in bijective correspondence with odd length alternating paths of source $((\langle a, d \rangle), q, \sigma, \vec{\alpha})$ for any a, d, σ . This is enough to show that M is acyclic against $\mathfrak{w}$ if and only if there are no alternating cycles between $\{M\}$ and $!_{[\mu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$. Moreover, this is true for any word $\mathfrak{w}$ and any $\mu \in I_{|\mathfrak{w}|}$. Thus M is acyclic if and only if $\{M\}$ is. $\square$

Definition 86. We define the test $\mathcal{T}_+^{\text{det}}$ as the single project $\mathfrak{o}^{(r)} \otimes 1/2_{\langle a \rangle}, \zeta \neq 0$.

Proposition 87. Suppose the map m is chosen so that $\llbracket A, B \rrbracket_m \neq \infty, 0$ as soon as there is a cycle between A and B of weight different from 1. A project $(0, M)$ of support $\langle a, r \rangle$ is orthogonal to $\mathcal{T}_+^{\text{det}}$ if and only if there is a sequence $q_1, \dots, q_k$ of different elements in D^M and a family of edges from $\langle a \rangle, q_i$ to $\langle a \rangle, q_{i+1}$ (with $q_{k+1} = q_1$) in M .

Proof. We compute $\llbracket M, \text{Id}_{\langle x \rangle} \rrbracket_m \neq 0, \infty$. If there is such a family of edges in M , then it creates a cycle with $\text{Id}_{\langle x \rangle}$ of weight different from 1. As a consequence the measurement $\llbracket M, \text{Id}_{\langle x \rangle} \rrbracket_m$ is different from 0 and ∞ , i.e. $(0, M)$ is orthogonal to $\mathcal{T}_+^{\text{det}}$. If there are no such family in M , then the measurement $\llbracket M, \text{Id}_{\langle x \rangle} \rrbracket_m$ is equal to 0 since the existence of an alternating cycle between M and Id_r implies the existence of such a family of edges. In that case, the projects $(0, M)$ and $\mathcal{T}_+^{\text{det}}$ cannot be orthogonal. $\square$

Theorem 88. Any language computed by an acyclic k -head automaton is computed by a $(\{1\}, m_k)$ -machine in $! \text{Words}_{\{0,1\}}^{(2)} \multimap \text{Bool}$.

Proof. The proof consists solely in arranging together previous propositions. Take $\mathcal{L}$ a language computed by an acyclic k -head automaton M . Then the graphing $\{M\}$ is an acyclic $(\{1\}, m_k)$ -machine from Proposition 85. From Proposition 87 we deduce that, for any word $\mathfrak{w}$, the project $(0, \{M\}) :: !_{[\mu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$ is orthogonal to $\mathcal{T}_+^{\text{det}}$ if and only if there exists a path from $\langle a \rangle$ to itself in $(0, \{M\}) :: !_{[\mu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$. By Corollary 83, such a path exists if and if M accepts the input $\mathfrak{w}$. I.e. $\mathcal{L} = \mathcal{L}^{\mathcal{T}_+^{\text{det}}}(M)$. $\square$

4.5. Machines as Automata. The following is a technical lemma that uses some particular properties of the microcosm $\mathfrak{m}_{\infty}$. This lemma is the equivalent, on our framework, to the so-called technical lemma which was essential in previous work involving operator algebras [AS14a, AS14b].

Lemma 89 (Technical Lemma). Let Ω be a monoid and $\mathfrak{M} = (0, M)$ be a $(\Omega, \mathfrak{m}_{\infty})$ -machine project. The computation of $\mathfrak{M}$ with a the exponentiation $!_{[\mu]}^{\Xi} \{\mathfrak{w}\}$ of a representation of a word $\mathfrak{w}$ is the realisation of a Ω -weighted finite graph.

Proof. The proof of this lemma is based on the finiteness of $(\Omega, \mathfrak{m}_{\infty})$ -machines. Since M is a finite graphing, there exists an integer N such that M is a $(\Omega, \mathfrak{m}_N)$ -machine. We are thus left to prove the result for M a $(\Omega, \mathfrak{m}_N)$ -machine. We now pick a word $\mathfrak{w} \in \Sigma^*$, write k the length of $\mathfrak{w}$ and $(0, W_{\mathfrak{w}})$ the project $!_{[\mu]}^{\Xi} \{\mathfrak{w}\}$. Let us remark that all maps realising edges in M or in $W_{\mathfrak{w}}$ are of the form $\phi \times \text{Id}_{\times_{i=N+1}^{\infty} [0,1]}$. We can therefore consider that the underlying space is $[0, 1]^N$. Moreover, the maps ϕ in this decomposition act either as permutations over copies of $[0, 1]$ (realisers of edges of M) or as permutations over a decomposition of $[0, 1]$ into

k intervals (the length of $\mathfrak{w}$). Consequently, all realisers act as permutations over the set of N -cubes $\{\times_{i=1}^N [k_i/k, (k_i+1)/k] \mid 0 \leq k_i \leq k-1\}$, i.e. their restrictions to N -cubes are identities.

Consequently, one can build two thick graphs $\bar{M}$ and $\bar{W}_{\mathfrak{w}}$ over the set of vertices $\Sigma^{\mathbb{N}} \times \{\times_{i=1}^N [k_i/k, (k_i+1)/k] \mid 0 \leq k_i \leq k-1\}$ as follows. There is an edge in $\bar{M}$ of source $(s, (k_i)_{i=1}^N, d)$ to $(s', (k'_i)_{i=1}^N, d')$ if and only if there is an edge in M of source $\langle s \rangle \times \{d\}$ and target $\langle s' \rangle \times \{d'\}$ whose realisation send the N -cube $\times_{i=1}^N [k_i/k, (k_i+1)/k]$ onto the N -cube $\times_{i=1}^N [k'_i/k, (k'_i+1)/k]$. There is an edge in $\bar{W}_{\mathfrak{w}}$ of source $(s, (k_i)_{i=1}^N, d)$ to $(s', (k'_i)_{i=1}^N, d')$ if and only if $d = d'$, $k_i = k'_i$ for $i \geq 2$ and there is an edge in $W_{\mathfrak{w}}$ of source $\langle s \rangle \times [k_1/k, (k_1+1)/k] \times [0, 1]^{\mathbb{N}}$ and target $\langle s' \rangle \times [k'_1/k, (k'_1+1)/k] \times [0, 1]^{\mathbb{N}}$.

Then, checking the existence of an alternating path between M and $W_{\mathfrak{w}}$ turns out to be equivalent to the existence of an alternating path between $\bar{M}$ and $\bar{W}_{\mathfrak{w}}$. $\square$

This technical lemma could be used, as in previous work [AS14a, AS14b], to prove that the language computed by a $(\Omega, \mathfrak{m}_{\infty})$ -machine M is LOGSPACE computable. Indeed, using this lemma, the problem of deciding whether a word is accepted by M can be reduced to the reachability problem in a graph (i.e. we check the existence of paths alternating between the graphs $\bar{M}$ and $\bar{W}_{\mathfrak{w}}$). We will, however, prove a more precise result in this paper, namely that a language computed by a $(\Omega, \mathfrak{m}_N)$ -machine M can be computed by a N -head automata. This implies in particular that languages computed by $(\Omega, \mathfrak{m}_{\infty})$ -machines are LOGSPACE computable, but this provides an exact correspondence between the hierarchy of microcosms $\mathfrak{m}_i$ and the classes 2NFA(i). It turns out that the technical lemma we just shown is not used in the following proofs. We chose to include it anyway in order to stress the relationship between our results and previous work [AS14a, AS14b].

First, we prove a result which holds in full generality, namely that if m is a set of measurable maps generating a microcosm $\mathfrak{m}$, then all $(\Omega, \mathfrak{m})$ -machine can be simulated by a (Ω, m) -machine.

Definition 90. A $(\Omega, \mathfrak{m})$ -machine M is *essential* w.r.t. a generating set G of the microcosm $\mathfrak{m}$ if every edge $e \in E^M$ is realised by a restriction of a map in G .

Theorem 91. Let G be a set of measurable maps, $\mathfrak{m}$ the microcosm generated by m . For all $(\Omega, \mathfrak{m})$ -machine M , there exists an essential (Ω, m) -machine $\bar{M}$ w.r.t. G such that, for all test $\mathcal{T}$, $\mathcal{L}^{\mathcal{T}}(M) = \mathcal{L}^{\mathcal{T}}(\bar{M})$.

Proof. Let M be a machine, and e an edge realised by ϕ which is not an element of the generating set G . Then ϕ can be written as a composite $\phi = g_p \circ g_{p-1} \circ \dots \circ g_1$ where all elements g_i are in G . Now, we will show that this edge can be replaced by a series of edges realised by the maps g_i through the consideration of additional states. Let us write down (s, t, S, d) and (s', t', T, d') the respective source and target of the edge e , where $s, s' \in \Sigma_{\star}$, $t, t' \in \{\text{in}, \text{out}\}$ and d, d' are states. We consider a set of fresh state names new_i for $1 \leq i \leq 2p-2$, and consider the machine $M^{(e)}$ obtained from M by adding these new states, removing the edge e , and adding the following edges:

- an edge e_1 of source (s, t, S, d) and target $(s, \bar{t}, g_1(S), \text{new}_1)$ realised by g_1 ;
- more generally, edges e_{i+1} ($i = 1, 2, \dots, p-1$) of source $(s, t, g_i \circ \dots \circ g_1(S), \text{new}_{2i})$ and target $(s, \bar{t}, g_{i+1} \circ g_i \circ \dots \circ g_1(S), \text{new}_{2i+1})$ realised by g_{i+1} ;
- edges $e_{\text{rev}_i}^a$, for all symbols $a \in \Sigma_{\star}$, of source $(a, t, g_{i+1} \circ g_i \circ \dots \circ g_1(S), \text{new}_{2i-1})$ and target $(a, \bar{t}, g_{i+1} \circ g_i \circ \dots \circ g_1(S), \text{new}_{2i})$ realised by a translation;
- Finally, an edge e_p of source $(s, t, g_{p-1} \circ \dots \circ g_1(S), \text{new}_{2p-2})$ and target (s', t', T, d') realised by g_p .

Now, we can show that the languages of M and $M^{(e)}$ are the same based on the fact that in $M^{(e)}$ we are stalling the computation by going back and forth between a symbol and its preceding symbol until we have computed all g_i steps.¹³ The consequence is that there is a bijective correspondence between paths alternating between M and W and paths alternating between $M^{(e)}$ and W , i.e. up to extending the set of states, $M :: W$ and $M^{(e)} :: W$ are equal. This implies that $\mathcal{L}^{\mathcal{T}}(M) = \mathcal{L}^{\mathcal{T}}(M^{(e)})$.

Now, writing $e_1, e_2, \dots, e_l$ the set of all edges in M that are realised by an element $\phi \notin G$, we can build the sequence of machines $M_0 = M$ and $M_{i+1} = M_i^{(e_{i+1})}$ and define $\tilde{M}$ as M_l . Then obviously $\tilde{M}$ is essential w.r.t. G , and satisfies $\mathcal{L}^{\mathcal{T}}(M) = \mathcal{L}^{\mathcal{T}}(\tilde{M})$. $\square$

Using this result, we can show the wanted inclusion (i.e. completeness of the model). For this we consider an essential $(\{1\}, m_N)$ -machine G w.r.t. the generating set Γ consisting in the microcosm m_1 and the maps s_i obtained from the transposition exchanging 1 and i . We will construct an automaton $[G]$ that computes the same language as G .

We construct the automaton $[G]$ as follows. Let Q denote the dialect of the thick graphing M . We denote by $\mathcal{J}$ the set of vertices $(\langle a \rangle, q)$, with $q \in Q$, which are both a source and a target of edges in M . If $\mathcal{J}$ is empty, then $\mathcal{L}^{\mathcal{T}}(M) = \emptyset$ which is clearly computed by an automaton with at most N heads. We now suppose that $\mathcal{J} \neq \emptyset$ and fix a total ordering on $\mathcal{J}$. We will build an automaton $[G]$ whose set of states is

$$Q \times \mathfrak{G}_k \times \mathcal{P}(\mathcal{J})^2 \times \{\star, 0, 1\} \times \{\text{in}, \text{out}\} \uplus \mathbb{I}$$

The permutations will be used to keep track of the exchanges of heads during the computation. The subsets of $\mathcal{P}(\mathcal{J})$ are used to keep track of the edges in $\mathcal{J}$ that have not been visited yet. The couples $(s, d) \in \{\star, 0, 1\} \times \{\text{in}, \text{out}\}$ will be used to remember important informations. Lastly, the disjoint set of states $\mathbb{I}$ will be used to implement a *reinitialisation procedure* which moves all heads to the start of the list. Notice that since M was supposed essential, all edges are realised by a simple transposition $\tau_{1,j}$ composed with a bijection on $\mathbf{N}$; we abusively say that the edge is realised by the transposition to lighten the definition of the automaton.

An edge e in M , of source $(\langle a \rangle, q)$ with $q \in \mathcal{J}$ and target $(\langle s', d' \rangle, q')$ realised by $\tau_{1,j}$ is represented by the family of transitions $(\vec{a}, (q, \sigma, P, P')) \rightarrow (\sigma(j), d', (q', \tau_{1,j} \circ \sigma, P, P' \cup \{q\}))$ for $\vec{a}$ such that $a_{\sigma(j)} = s'$ and P' not containing q .

An edge e in M , of source $(\langle s, d \rangle, q)$ and target $(\langle s', d' \rangle, q')$ realised by $\tau_{1,j}$ is represented by the family of transitions $(\vec{a}, (q, \sigma, P, P')) \rightarrow (\sigma(j), d', (q', \tau_{1,j} \circ \sigma, P, P'))$ for $\vec{a}$ such that $a_{\sigma(j)} = s'$ and $a_{\sigma(1)} = s$.

An edge e in M , of source $(\langle s, d \rangle, q)$ and target $(\langle a \rangle, q)$ with $q \in \mathcal{J}$ realised by $\tau_{1,j}$ is represented by:

- the family of transitions $(\vec{a}, (q, \sigma, P, P')) \rightarrow (\sigma(j), d', (q', \tau_{1,j} \circ \sigma, P, P'))$ for $\vec{a}$ such that $a_{\sigma(1)} = s$ and $a_{\sigma(j)} = s'$ and P' not containing q ;
- the family of transitions $(\vec{a}, (q, \sigma, P, P')) \rightarrow \text{accept}$ for $\vec{a}$ such that $a_{\sigma(1)} = s$ and $a_{\sigma(j)} = s'$ and P' containing q ;

The transition relation thus obtained is then completed with both rejecting transitions and a reinitialisation procedure for heads that keeps track of the set P :

- the set $\mathbb{I}$ is chosen equal to $\{l_i \mid 0 \leq i \leq N\} \times \mathcal{P}(\mathcal{J})$;
- the transitions considered are:

¹³Let us notice that in the case of one-way integers, the stalling is represented in another way, but can be represented as well and the same result holds.

- a family of transitions starting a reinitialisation: $(\vec{a}, (q, \sigma, P, P')) \rightarrow (\iota_1, P)$ for every $(\vec{a}, (q, \sigma, P, P'))$ which is not in the domain of the transition $\rightarrow$ defined above and such that $P \neq \emptyset$;
- the transitions for the reinitialisation itself:

$$\underbrace{(\star, \dots, \star)}_{i-1}, a, s_{i+1}, \dots, s_N; \iota_i, P \rightarrow \begin{cases} (i, \text{out}; \iota_i, P) & \text{when } a \neq \star \\ (i+1, \text{out}; \iota_{i+1}, P) & \text{when } a = \star \end{cases}$$

- the transitions that restart the computation $(\vec{\star}; \iota_{N+1}, P) \rightarrow (e; \inf P, \text{Id}, P - \inf P, \emptyset)$.
- a family of transitions representing rejection $(\vec{a}, (q, \sigma, P, P')) \rightarrow (\iota_1, P)$ for every $(\vec{a}, (q, \sigma, P, P'))$ which is not in the domain of the transition $\rightarrow$ defined above and such that $P = \emptyset$.

Definition 92. For any essential $(\{1\}, m_N)$ -machine G w.r.t. the generating set Γ , we denote $[G]$ the N -head automaton described above.

Proposition 93. Let G be an essential $(\{1\}, m_N)$ -machine w.r.t. the generating set Γ and $!_{[\mu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$ be a word representation of the word $\mathfrak{w}$. There is an alternating cycle between $G :: !_{[\mu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$ and $\text{Id}_{\langle a \rangle}$ if and only if the automaton $[G]$ accepts when given $\mathfrak{w}$ as input.

Proof. With the definitions above, one can notice that there is an alternating path between G and a representation of the word $\mathfrak{w}$ whose source and target are $(\langle a \rangle, q) \in \mathcal{J}$ and $(\langle a \rangle, q') \in \mathcal{J}$ respectively if and only if there is a computation trace of the constructed automaton $[M]$ on input $\mathfrak{w}$ starting at configuration $(0, 0, \dots, 0, (q, \text{Id}, P \cup \{(\langle a \rangle, q')\}), P', s, d)$ and ending at $(0, 0, \dots, 0, (q', \text{Id}, P, P' \cup \{(\langle a \rangle, q')\}), s, d)$. Now, if there is a cycle between $G :: !_{[\mu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$ and $\text{Id}_{\langle a \rangle}$, it goes through a sequence of elements of $\mathcal{J}$, and one can show the existence of an accepting computational trace of $[M]$ on input $\mathfrak{w}$ starting at configuration $(0, 0, \dots, 0, (q, \text{Id}, P \cup \{(\langle a \rangle, q')\}), P', s, d)$. A last argument shows that if an element of $\mathcal{J}$ which is not part of the cycle is chosen for starting the computation, then either it was part of another cycle (and then computation stops, but the things are orthogonal so that's ok), or it is not and then we restart the computation with a new value in $\mathcal{J}$. Since $\mathcal{J}$ is finite, we are bound to go through the cycle at some point. $\square$

Proposition 94. Let G be an essential $(\{1\}, m_N)$ -machine w.r.t. the generating set Γ . If G is acyclic, so is $[G]$.

Theorem 95. Any language computed by a $(\{1\}, m_N)$ -machine in $!_{\{0,1\}} \mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{Bool}$ is computed by an acyclic N -head automaton.

Proof. Pick a language $\mathcal{L}$ for which there exists a $(\{1\}, m_N)$ -machine M with $\mathcal{L} = \mathcal{L}_{+}^{\mathcal{T}^{\text{det}}}(M)$. We can suppose by Theorem 91 that the machine M is *essential* w.r.t. the generating set consisting in the microcosm m_1 and the maps s_i obtained from the transposition exchanging 1 and i . Now, by Corollary 83 the machine M accepts a word $\mathfrak{w}$ if and only if there is an alternating cycle between $M :: !_{[\mu]}^{\Xi} \bar{W}_{\mathfrak{w}}^{(2)}$ and $\text{Id}_{\langle a \rangle}$. This is equivalent, by Proposition 93, to the fact that $[M]$ accepts when given $\mathfrak{w}$ as input. Thus $\mathcal{L}$ is the language computed by $[M]$. Moreover, the machine $[M]$ is acyclic by Proposition 94. $\square$

5. VARIATIONS

In this section, we show how the results exposed in the preceding section can be adapted to obtain similar results for non-deterministic and probabilistic computation, as well as deal

with one-way automata. The first two subsections are devoted to non-deterministic and probabilistic computation respectively. We show that the preceding characterisation result lifts to these frameworks, obtaining in particular characterisations of the classes NLOGSPACE, CONLOGSPACE and PLOGSPACE. The last section introduces the type of one-way representations of integers, and shows that one can characterise the sets of languages accepted by one-way multihead automata.

5.1. The Non-Deterministic Hierarchies. The adaptation to non-determinism is quite straightforward. Indeed, the translations given in the previous section, from automata to machines and back, define translations between non-deterministic k -head automata and non-deterministic (Ω, m_k) -machines.

Definition 96. We define the test $\mathcal{T}_-^{\text{det}}$ as the set consisting of the projects $\partial \text{ai}_\zeta^{(a)} \otimes 1_{\langle r \rangle}$, $\zeta \neq 0$.

Proposition 97. Suppose the map m is chosen so that $\llbracket A, B \rrbracket_m \neq 0$ as soon as there is a cycle between A and B . A project $(0, M)$ of support $\langle a, r \rangle$ is orthogonal to $\mathcal{T}_-^{\text{det}}$ if and only if there are no edges from $\langle r \rangle$ to itself in M .

Proof. Suppose that $(0, M)$ is orthogonal to $\mathcal{T}_-^{\text{det}}$. Then for all $\zeta \neq 0$, $\zeta + \llbracket M, \text{Id}_{\langle r \rangle} \rrbracket_m \neq 0, \infty$. If there is an edge from $\langle r \rangle$ to itself in M , then it creates a cycle with $\text{Id}_{\langle r \rangle}$, and the measurement $\llbracket M, \text{Id}_{\langle r \rangle} \rrbracket_m$ is different from 0, leading to a contradiction as it is either ∞ or can be canceled with the an edaquate choice of ζ .

Conversely, if there are no edges from $\langle r \rangle$ to itself in M , then the measurement $\llbracket M, \text{Id}_{\langle r \rangle} \rrbracket_m$ is equal to 0. Hence for all $\zeta \neq 0$, $\zeta + \llbracket M, \text{Id}_{\langle r \rangle} \rrbracket_m = \zeta \neq 0, \infty$, i.e. $(0, M)$ is orthogonal to $\mathcal{T}_-^{\text{det}}$. $\square$

5.2. The Probabilistic Hierarchy. We now consider a variant of the classical notion of *probabilistic* two-way multihead finite automata [?] obtained from the same choices in design that in the non-probabilistic case. Once again, these choices have no effect on the sets of languages recognized. For a set S , we denote by $\mathcal{P}(S)$ the set of probability distributions on S .

Definition 98. A *probabilistic two-way multihead automata* M with k heads is defined as a tuple $(\Sigma, Q, \rightarrow)$, where $\rightarrow: ((\Sigma \cup \{\star_{\text{out}}, \star_{\text{in}}\})^k \times Q) \rightarrow \mathcal{P}((\{1, \dots, k\} \times \{\text{in}, \text{out}\}) \times Q)$ is the *transition* of M . The automaton M is *deterministic* when the relation $\rightarrow$ is functional.

The set of two-way multihead automata with k heads is written $\mathbf{2nfa}(k)$, and the set of all two-way multihead automata $\cup_{k \geq 1} \mathbf{2nfa}(k)$ is denoted by $\mathbf{2nfa}$. The set of deterministic two-way multihead automata with k heads is written $\mathbf{2dfa}(k)$, and the set of all deterministic two-way multihead automata $\cup_{k \geq 1} \mathbf{2dfa}(k)$ is denoted by $\mathbf{2dfa}$.

Definition 99. We denote, for all integer k , by $\mathbf{2DPFA}(k)$ the set of languages accepted by probabilistic multihead automata in $\mathbf{2pdfa}(k)$.

Remark 100. The set of languages $\text{STOCHASTIC} = \mathbf{2DPFA}(1)$ is usually called the set of *stochastic languages*.

Theorem 101. The set of languages recognized by all probabilistic multihead automata is equal to PLOGSPACE:

$$\cup_{k \geq 1} \mathbf{2DPFA}(k) = \text{PLOGSPACE}$$

Now, the translations between non-deterministic k -head automata and non-deterministic $(\Omega, \mathfrak{m}_k)$ -machines provides translations between probabilistic variants of those by simply making the probabilities of a transition in an automaton correspond to the weights of the corresponding edge in the graphing. The only important point is again the question of testing, i.e. does there exist a test corresponding to the unbounded error probabilistic acceptance? The answer is positive, as already mentioned in previous sections.

Definition 102. We define the test $\mathcal{T}^p$ as the set of all projects $(\zeta, 1_{\langle a \rangle})$ for $0 < \zeta < \log(2)$.

Proposition 103. Suppose that the measurement has the contraction property, i.e. $\llbracket F, G \rrbracket_m = \llbracket \angle(F), G \rrbracket_m$ for all F, G . If $(0, M)$ is orthogonal to $\mathcal{T}^p$, then $\angle(M)$ has an edge from $\langle a \rangle$ to itself whose weight is strictly greater than $1/2$.

5.3. One-Way Automata. We will here explain how one-way automata can be interpreted in the models. This is done by considering the same notion of machines but a different representation of words. We first define formally the graph representing a word over a chosen alphabet Σ .

Definition 104. We define the graphs $W_{\mathfrak{w}}^{(1)}$ representing a word $\mathfrak{w} = \star a_1 a_2 \dots a_k \in \Sigma^*$:

$$\begin{aligned} V^{W_{\mathfrak{w}}^{(1)}} &= \Sigma^{\mathbb{N}} \\ D^{W_{\mathfrak{w}}^{(1)}} &= \{0, 1, \dots, k, k+1\} \\ E^{W_{\mathfrak{w}}^{(1)}} &= \{r, l\} \times \{0, \dots, k\} \\ s^{W_{\mathfrak{w}}^{(1)}} &= (r, i) \mapsto (a_i, \text{out}, i) \\ &\quad (l, i) \mapsto (a_i, \text{in}, i) \\ t^{W_{\mathfrak{w}}^{(1)}} &= (r, i) \mapsto (a_{i+1}, \text{in}, i+1) \\ &\quad (l, i) \mapsto (a_i, \text{in}, i) \end{aligned}$$

As in the case of two-way representations, one defines the corresponding graphing $\bar{W}_{\mathfrak{w}}^{(1)}$.

We stress that the one-way representation $W_{\mathfrak{w}}^{(1)}$ of a word $\mathfrak{w}$ is *not* obtained by simply replacing two-way edges by one-way edges in the two-way representation $W_{\mathfrak{w}}^{(2)}$. Indeed, we also added loops in the graph that allows for the machines to *stall*, i.e. machines will have the possibility of not moving heads when performing a transition. One could also study the notion of *strict one-way representation*, i.e. the graph $W_{\mathfrak{w}}^{(1)}$ without the added loops; this notion of integers should allow the characterisation of machines which move exactly one of their head at each transition step.

Remark 105. Although we do not want to go into the details of this notion of computation, let us remark that – with a fixed number of heads – the machines that have the possibility to stall should be strictly more expressive than those that do not. For instance, if M is a one-way machine that can stall, it is not clear that one can define a non-stalling one-way machine M' with the same number of heads as M and recognizing the same language. Indeed, even considering an additional head used to stall (i.e. one moves this additional head each time M stalls) seems insufficient since this additional head can be used to stall only a finite number of times (the size of the input). However, since the number of stalling transitions of M is bounded by a polynomial in the size of the input, one could show that there exists a non-stalling machine M' with sufficiently many additional “stalling heads” that recognizes the same language as M . As a consequence, one knows that the class of languages accepted by

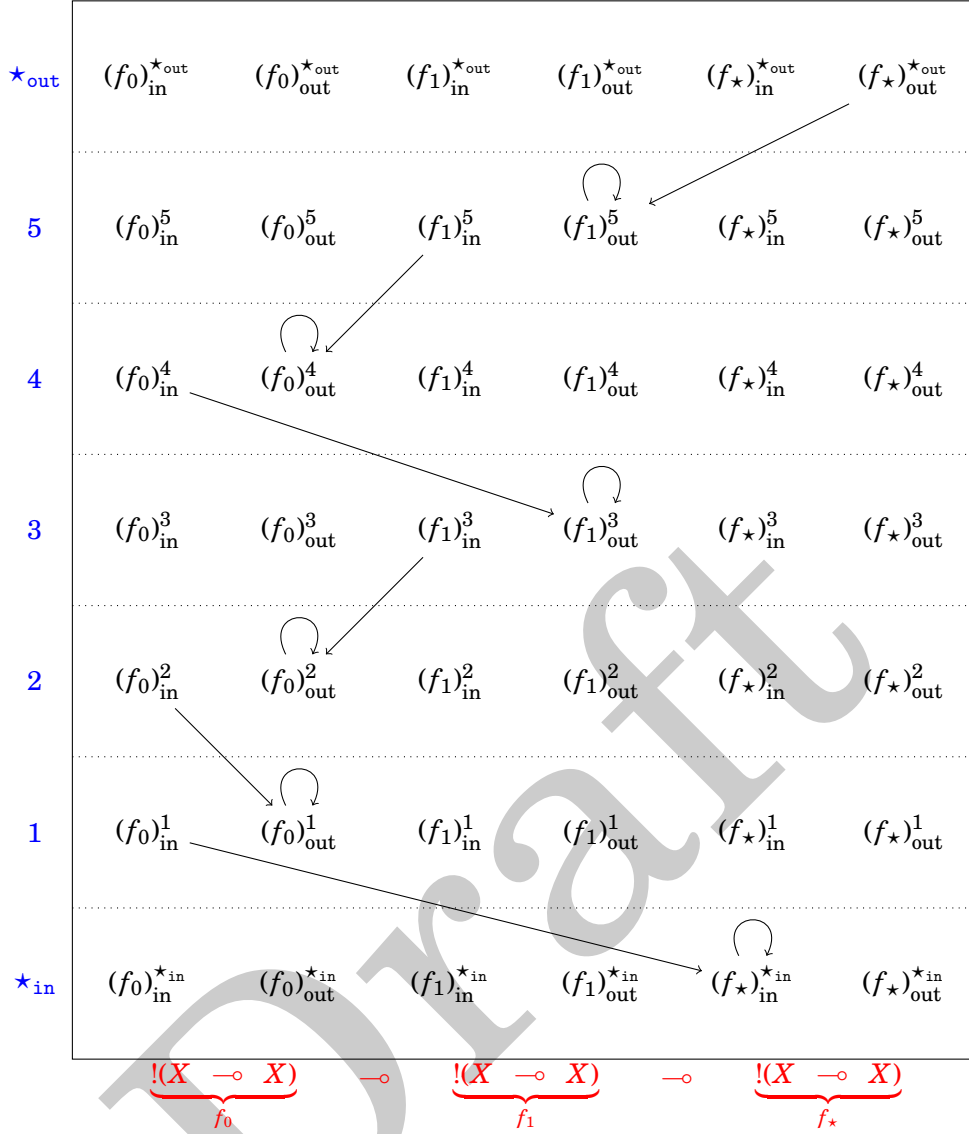


Figure 7: Example: the one-way representation $W_{\star 00101}^{(1)}$ of the binary word $\star 00101$. The dialect is represented on a vertical scale.

stalling one-way machines is equal to the class of languages accepted by non-stalling one-way machines, even though the same result does not hold for a fixed number of heads. Finally, let us notice that, in the case of two-way machines, the possibility of stalling does not change anything as a machine with k heads can be mimicked by a non-stalling machine with k heads by simply adding states: stalling is then represented as making one move to the left followed by one move to the right.

6. CONCLUSION AND PERSPECTIVES

6.1. Relations with the word problem. This paper was concerned mainly with the possibility of capturing different complexity classes by playing on one particular parameter of the Interaction graphs construction [Sei14e]. Although a few examples were developed where one considers modifications of other parameters (for extensions to probabilistic and/or quantum computing), we believe that these examples are only a tiny fraction of what could be done by playing on the monoid of weights and the measuring map m defining the orthogonality. In particular, these parameters may allow for relating our setting to the word problem for monoids and/or groups. As a simple example, one might consider the microcosm characterising regular languages. As shown in this paper, the elements of $!\mathbf{Words}_2^{(2)} \multimap \mathbf{Bool}$ can be identified with two-way automata. If one extends this model by considering the monoid of weights to be the free group $\mathbb{F}_1$ over one element – say a , and considers the measurable map $m : \Omega \rightarrow \mathbf{R}_{\geq 0} \cup \{\infty\}$, $a \neq 1 \mapsto 0$ and $1 \mapsto 1$, then the elements in $!\mathbf{Words}_2^{(2)} \multimap \mathbf{Bool}$ become assimilable to two-way automata with a counter, where such a machine accepts only when the counter is equal to 0. This simple example shows how one can encode some specific machine constructs as algebraic equalities which can be incorporated in the model through the parameter Ω and m . It is obvious that more involved examples can be related to the word problem, or variants of it. For instance, one can consider the monoid $\mathcal{S}$ generated by five elements a, b, c, d, e extended by an absorbing element 0, and subject to the following equalities:

$$ac = a, bc = 0, ad = 0, bd = b, ae = be = 1$$

One will check that this algebraic structure can be used to encode a simple stack containing binary words, with the possibility of testing the last value stacked, since for instance $w.c = 0$ if $w = w'.b$ and $w.c = w$ if $w = w'.a$, and erasing the last value stacked, since $w'.a.e = w'.b.e$. We now consider the model $\mathbb{M}_m^{\text{det}}[\mathcal{S}, m_\infty]$ defined by choosing the measuring map $m : \mathcal{S} \rightarrow \mathbf{R}_{\geq 0} \cup \{\infty\}$ defined by $m(w) = 0$ unless $w = 1$ in which case $m(w) = 1$. Elements of $\mathbf{Words}_2^{(2)} \multimap \mathbf{Bool}$ in this model correspond to two-way multihead automata with a binary stack, a notion of machines known to characterise the class PTIME. Hence, the type of binary predicates in this model will characterise PTIME, showing the word problem for $\mathcal{S}$ to be decidable in polynomial time.

6.2. Extensions to other Classes.

6.3. Cohomological Invariants and Complexity. From the author's point of view, the most exciting perspective is the possibility to relate complexity classes with homological and/or cohomological invariants. Indeed, we showed in this paper a partial correspondence between microcosms and complexity classes. In this particular case, all microcosms are sub-monoids of the monoid of measure-preserving maps. In this respect, they generate a measured equivalence relation on the underlying measure space $\mathbf{X}$. This leads to the natural idea that invariants for measured equivalence relations, such as ℓ^2 -Betti numbers [Gab02] might be used to distinguish between complexity classes. In particular, the microcosms considered here are obtained as realisations of measure-preserving group actions. It seems that in this special case, the complexity classes are homotopy invariants, i.e. if two G -spaces are homotopy equivalent, then the associated complexity classes are equal. The hope is that such invariants would provide a new proof method for separation results, something actually lacking from the complexity-theorist toolbox as all known methods fall into the notion of *natural proofs* defined and shown useless for separation results twenty years ago [RR97].

To follow this approach, it is however necessary to obtain characterisations of complexity classes based solely upon the choice of a microcosm. To be more precise, if new methods for separation of complexity classes could be obtained from the use of cohomological invariants in this way, one could use these methods to separate classes characterised by different microcosms and the same monoid of weights¹⁴. For instance, the characterisation of LOGSPACE developed in this paper and the characterisation of PTIME sketched above could not lead to a separation result using such methods since they differ only on their monoid of weights. It is therefore important to look for characterisations of complexity classes based solely upon the change of microcosms. Moreover, since cohomological invariants exists for the measure-preserving case one should look for characterisations by microcosms of measure-preserving maps whenever possible, since non-measure-preserving microcosms would need the definition of a cohomological theory in this extending setting. Although we do not believe this to be impossible, we think better to try and restrict ourselves to the well-studied and simpler case of measure-preserving maps which, in our opinion, is already rich enough to provide many interesting characterisations.

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.
- [Ada90] Scot Adams. Trees and amenable equivalence relations. *Ergodic Theory and Dynamical Systems*, 10:1–14, 1990.
- [AS14a] Clément Aubert and Thomas Seiller. Characterizing co-nl by a group action. *Mathematical Structures in Computer Science*, 2014. to appear.
- [AS14b] Clément Aubert and Thomas Seiller. Logarithmic space and permutations. *Information and Computation*, 2014. to appear.
- [BP01] Patrick Baillot and Marco Pedicini. Elementary complexity and geometry of interaction. *Fundamenta Informaticae*, 45(1-2):1–31, 2001.
- [Car04] Elie Cartan. Sur la structure des groupes infinis de transformations. *Annales Scientifiques de l'École Normale Supérieure*, 21:153–206, 1904.
- [Car09] Elie Cartan. Les groupes de transformations continus, infinis, simples. *Annales Scientifiques de l'École Normale Supérieure*, 26:93–161, 1909.
- [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.
- [Dan90] Vincent Danos. *La Logique Linéaire Appliquée à l'Étude de Divers Processus de Normalisation (principalement du λ -calcul)*. PhD thesis, Paris VII University, 1990.
- [DJ03] Vincent Danos and Jean-Baptiste Joinet. Linear logic & elementary time. *Information and Computation*, 183(1):123–137, 2003.
- [FK52] Bent Fuglede and Richard V. Kadison. Determinant theory in finite factors. *Annals of Mathematics*, 56(2), 1952.
- [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.
- [Gir87a] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987.

¹⁴Or at least an inclusion of the monoid of weights "in the right direction".

- [Gir87b] Jean-Yves Girard. Multiplicatives. In Lolli, editor, *Logic and Computer Science : New Trends and Applications*, pages 11–34, Torino, 1987. Università di Torino. Rendiconti del seminario matematico dell’università e politecnico di Torino, special issue 1987.
- [Gir88] Jean-Yves Girard. Normal functors, power series and λ -calculus. *Annals of Pure and Applied Logic*, 37(2):129–177, February 1988.
- [Gir89] Jean-Yves Girard. Geometry of interaction I: Interpretation of system F. In *In Proc. Logic Colloquium 88*, 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 ’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.
- [Gir13a] Jean-Yves Girard. Geometry of interaction vi: a blueprint for transcendental syntax. 2013.
- [Gir13b] Jean-Yves Girard. Three lightings of logic (invited talk). In Simona Ronchi Della Rocca, editor, *CSL*, volume 23 of *LIPICs*, pages 11–23. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
- [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.
- [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*, 1988.
- [KN96] S. Kobayashi and K. Nomizu. *Foundations of Differential Geometry*. Number vol. I in A Wiley Publication in Applied Statistics. Wiley, 1996.
- [Kri01] Jean-Louis Krivine. Typed lambda-calculus in classical zermelo-fraenkel set theory. *Archive for Mathematical Logic*, 40(3):189–205, 2001.
- [Kri09] Jean-Louis Krivine. Realizability in classical logic. *Panoramas et synthèses*, 27:197–229, 2009.
- [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.
- [Lev95] Gilbert Levitt. On the cost of generating an equivalence relation. *Ergodic Theory and Dynamical Systems*, 15:1173–1181, 1995.
- [Mon76] Buckhard Monien. Transformational methods and their application to complexity problems. *Acta Informatica*, 6:95–108, 1976.
- [NPS15] Alberto Naibo, Mattia Petrolo, and Thomas Seiller. *On the Computational Meaning of Axioms*. Logic, Epistemology, and the Unity of Science. Springer, 2015.
- [Rib07] Colin Riba. Strong normalization as safe interaction. In *LICS 2007 Proceedings*, 2007.
- [RR97] A. A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55, 1997.
- [RS59] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.
- [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 : géométrie de l’interaction et complexité*. PhD thesis, Université Aix-Marseille, 2012.
- [Sei14a] Thomas Seiller. A correspondence between maximal abelian sub-algebras 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.
- [Sei15a] Thomas Seiller. Interaction graphs: Full linear logic. *Submitted*, 2015.
- [Sei15b] Thomas Seiller. Towards a *Complexity-through-Realizability* theory. *Submitted*, 2015.
- [Sze87] R. Szelepcényi. The method of forced enumeration for nondeterministic automata. *Bulletin of the EATCS*, 33, 1987.

Draft