uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Brian Redmond

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Mathematics)

GRADE / DEGREE

Department of Mathematics and Statistics

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Multiplexor Categories and Models of Soft Linear Logic

TITRE DE LA THÈSE / TITLE OF THESIS

Richard Blute

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Pieter Hofstra

Paul-Eugène Parent

David Howe

Philip Scott

Harry Mairson

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Multiplexor Categories and Models of Soft Linear Logic

By
Brian F. Redmond
January 2008

A Thesis submitted to the
School of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of
Ph.D. in Mathematics[1]

[1]The Ph.D. Program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

Your file Votre référence
ISBN: 978-0-494-41640-2
Our file Notre référence
ISBN: 978-0-494-41640-2

Abstract

This thesis furthers our semantical understanding of polynomial time complexity by clarifying the semantical status of Lafont's soft linear logic, a logical system complete for polynomial time computation. We shall see that soft linear logic is ideal for this purpose because it possesses a very natural and simple mathematical interpretation. To begin, we introduce the notion of a *multiplexor category*, which is the categorical interpretation of soft linear logic. We show that a multiplexor category provides a denotational semantics for soft linear logic and that the exponential operator can be interpreted canonically as a certain type of limit. This leads to a large class of models and motivates us to introduce a new class of AJM games which we shall call *noetherian*. Such games may be infinite, but do not have infinite length plays (i.e. there are no infinite increasing chains of plays in their game trees), hence total strategies will naturally compose.

Motivated by notions in complexity theory, we are led to introduce a finitary version of soft linear logic and a corresponding notion of finitary multiplexor category. One of the main results in this thesis is a categorical method for constructing a multiplexor category from a finitary one, called the S-construction, which leads to a stratified view of soft linear logic. This interpretation is inspired by the characterization of PTIME by uniformly polynomial circuits in complexity theory. Our construction has many applications. First, we use it to give a realizability model and a new proof that any algorithm representable in soft linear logic is polytime. Next, we utilize the S-construction to obtain a stratified finite game model of soft linear logic and prove a full-completeness result. Finally, we use it to give an alternative description of a model that has recently appeared in the literature. We also note that

the stratified interpretation allows for placing polynomial depth restrictions on the length of plays in our game model. Again, this is consistent with our circuit analogy.

We feel that this work provides an important first step towards a truly semantic perspective on PTIME complexity.

Acknowledgements

First and foremost, I would like to thank my supervisor Rick Blute for supporting this project. He has been an influential teacher and supportive advisor. I would also like to thank Phil Scott, Patrick Baillot, Olivier Laurent and Yves Lafont for their valuable input at various times throughout this project. As this thesis shows, I have been greatly influenced by all of their work.

Finally, I would like to sincerely thank the reviewers of this thesis for their helpful comments and suggestions.

Dedication

For Pia

Contents

Abstract ii

Acknowledgements iv

Dedication v

1 Introduction 1

1.1 Related Work 4

1.2 Our Contribution 8

2 Background 10

2.1 Intuitionistic Linear Logic 10

2.2 The Light Exponentials 12

2.3 Category Theory 13

3 Soft Linear Logic and PCL 24

3.1 Soft Linear Logic 25

3.2 Predicative Combinatory Logic 30

4 Multiplexor Categories 39

4.1 Multiplexor Categories 40

4.2 Multiplexor Functors and Transformations 44

4.3 Product Formulas 45

4.4 Coherence Spaces 51

5 The S-Construction **55**
5.1 Finitary Soft Linear Logic 56
5.2 The S-Construction 59
5.3 A Variant of the S-Construction 63
5.4 Obsessional Cliques 65
5.5 The S-Construction Revisited 67

6 A Realizability Model for $\mathbf{ISAL}_2$ **70**
6.1 Preliminaries 70
6.2 A Realizability Category $\mathcal{B}^s$ 71
6.3 Interpretation of ISAL_2 74
6.4 PTIME Correctness of ISLL_2 78

7 Stratified Finite MO-Games and FCT **83**
7.1 Games and Strategies 84
7.2 Network Protocol for IMSAL^n 92
7.3 IMSAL Full Completeness 93
7.4 Noetherian Games 97
7.5 Bounded Complexity Games 99

8 Conclusions and Further Work **100**

A Proof of Cut-Elimination for IMSLL **103**

B Table of Acronyms **110**

Bibliography **111**

Chapter 1

Introduction

Computational complexity is an important branch of computer science that measures the basic resources (time and memory, for example) needed for solving computational problems. Its aim is to understand why some problems are intrinsically harder to solve than others on a computer, and to classify them according to their degree of difficulty. Knowing the complexity of a problem often leads to a deeper understanding of it. Moreover, there is a practical interest, not only in applications where resources are limited, but in applications that require computationally hard problems, like cryptography, for example.

Computational problems are classified into complexity classes. These complexity classes are robust and in general do not depend on the particular model of computation that we are using. However, it will be convenient to fix a model of computation, and for this purpose, we shall use deterministic (single-tape) Turing machines (see [40], for example). Hence we are equating algorithms with their Turing machine representations. One method for measuring the computational complexity of a decidable problem is by measuring how much time it takes to solve it. The *time complexity* of a Turing machine M is given by a function $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of steps used by M on any input of length n (written in binary, say). As usual, we shall use *big-O* notation when describing the time complexity of a computational problem. We say $f(n) = O(g(n))$ if there exist positive integers c and n_0 such that for all $n \geq n_0$, $f(n) \leq cg(n)$.

We say an algorithm runs in *polynomial time* if it has time complexity $O(n^k)$ for some nonnegative integer k, and we let PTIME (or sometimes just P) denote the class of languages that are decidable by a polynomial time algorithm. This is a mathematically robust class that roughly corresponds to the class of problems realistically solvable on a computer. This makes it of great theoretical interest and the subject of much investigation.

PTIME has been extensively studied by various researchers and various characterizations have resulted. These results have collectively deepened our understanding of PTIME. We mention only a few of the contributions. In recursion theory, characterizations of PTIME have been obtained by restricting the primitive recursion scheme (see [13, 7], for example). We also mention characterizations in the framework of the typed lambda calculus (see [31], for example). A connection with proof theory was established in the work of Buss [12], who introduced first-order systems of bounded arithmetic characterizing PTIME. More recently, systems of linear logic have been introduced with modified exponentials that characterize PTIME (see [20], [18] and [28], for example). We will have more to say about the various linear logic characterizations later on. All of these approaches have this in common: they are logics or type systems which characterize complexity classes. The study of such systems has become a field in itself, called *implicit computational complexity.*

Linear logic was introduced by Girard in [17] and is based on the idea of resources. Its careful managing of resources makes it ideal for the study of implicit computational complexity. Removing Gentzen's structural rules of contraction and weakening leads to a linear (or multiplicative) conjunction $\otimes$ and a linear implication $\multimap$, and copying and erasing of resources is no longer permissable in general. The missing structural rules are transformed into logical rules for special connectives called *exponentials*, and the symbol ! is used to distinguish precisely when a resource is unlimited. Here, we will only be interested in the second-order intuitionistic system which is denoted by ILL_2[1]. Logically, we are thinking of proofs as programs and cut-elimination as program execution after Curry-Howard. Therefore, to characterize PTIME, we are interested in logical systems that 1) have a PTIME cut-elimination procedure, and

[1]See appendix B for a table of the linear logic acronyms used in this thesis.

2) can represent all PTIME algorithms. In this case, we say that the logical system is *complete* for polynomial time computation.

It is well-known that second-order intuitionistic logic can be embedded into ILL_2, which means that it is as strong as Girard's system **F** (see [19]). Moreover, cut-elimination is hyper-exponential. Therefore it gives little insight into polynomial time computation. But it was soon realized that contraction is the main cause of the exponential blow-up in complexity of cut-elimination, and that controlling contraction is an indirect way of controlling time complexity [20]. The logical status afforded contraction in linear logic gives us the flexibility needed to control it (and allows for the many interesting variations). In other words, by modifying the exponential connectives of linear logic, we can find logical subsystems that are intrinsically polynomial time. Various proposals were introduced, some of which will be discussed in more detail in later chapters. We will mention a few here. *Bounded linear logic* was introduced by Girard, Scedrov and Scott in [20]. It is a good system but has the drawback that polynomials appear explicitly in the types. Another system is Girard's *light linear logic* which has an extra exponential modality §, and a hybrid notion of sequent [18]. The latter is avoided in the affine version by Asperti in [5], where it was realized that unrestricted weakening does not affect the complexity bounds of the system, yet it greatly simplifies the syntax. *Elementary linear logic* is a variant of light linear logic which is complete for elementary recursive computation [18]. Lafont's *soft linear logic* is closer in spirit to BLL, but avoids the heavy syntax. It is the simplest of the above systems and will be studied in much detail in this thesis. These systems are often collectively referred to as "light" linear logics, and their exponential connectives are often called "light" also. We shall follow this practice here.

We feel that one of the most important problems connected with these systems is the absence of a (natural) semantic or mathematical interpretation. In several cases this has been done, and we will discuss several contributions in the next section. What we are striving for is a mathematical perspective on polynomial time computation. We are not interested here in characterizing provability, for which a phase semantics would suffice, but in modeling the proofs themselves. In this case, the appropriate aim is for a *full-completeness* theorem, a concept first formulated by Abramsky and

Jagadeesan in [3]. A categorical model $\mathcal{C}$ is *complete* if the hom-set $\mathcal{C}([\![A]\!], [\![B]\!])$ is non-empty iff $A \vdash B$ is provable in the logic. A model $\mathcal{C}$ is *fully-complete* if any $f : [\![A]\!] \to [\![B]\!]$ is the denotation of a proof of $A \vdash B$. Therefore, a fully-complete model of one of the above systems would be a "general semantics of polytime" [18]. Moreover, mathematical models might clarify syntactical issues, and lead to a more definitive syntax in several cases. We might also see common mathematical concepts emerge in previously unrelated systems. Ultimately, we are working towards a common semantical framework for these systems of linear logic.

1.1 Related Work

Several researchers have already begun to tackle the problem of finding a denotational semantics for light linear logics, and work is ongoing. The following list of results is not meant to be exhaustive; it is only meant to represent the contributions that have most influenced my own work.

Bounded linear logic: Here we mention the realizability model of BLL by Hofmann and Scott in [25], following ideas from [24]. In this work, the authors give a concrete categorical model of an affine version of BLL in which the morphisms are witnessed by polytime terms. The main result is a new proof that all numerical functions representable in that system are polytime which follows directly by soundness of the model. This was the main theorem in [20]. The authors also considered whether their techniques could be used for other light linear logics, but did not pursue the idea at length. We successfully adapted their techniques for modeling soft affine logic (see Chapter 6). However, we tend to agree with Abramsky [2] that this technique does not truly give a semantical viewpoint on polytime computation, since complexity constraints are simply built into the model as an assumption. What we are really looking for are *"key structural properties which characterize PTIME algorithmic processes"* [2].

Independently, Hofmann and Dal Lago used similar realizability techniques to model light, elementary, and soft affine logics and another system called LFPL using a common semantical framework [15]. The relationship to our work has yet to be

investigated.

Light linear logic: In [26], Kanovich, Okada and Scedrov give a fibred phase space model of light linear logic. Inspired by models of temporal logic, the authors introduce a stratified version of phase spaces. The syntactic notion of nesting of proof-boxes in LLL is reflected in the semantic notion of *stratification.* The main result is a strong completeness proof for LLL, which gives a purely semantic proof of cut-elimination. However, phase space models are ultimately about provability – they give no information about the proofs themselves.

Closer in spirit to our approach is Baillot's stratified coherence spaces model, which is a denotational semantics for both ELL and LLL [6]. Again, we see the notion of stratification appear, this time in the form of sequences of coherence spaces which eventually become stationary. Intuitively, the levels represent degrees of approximation. And a morphism/clique is required to satisfy a coherence condition at each level of approximation. In this model, the usual multiset bang construction in coherence spaces is used to model the exponential connective !, and the exponential connective § is interpreted as a type of "shift" operator on sequences. This setting gives a model ELL, but it allows for a multifunctorial !, which is forbidden in LLL. By introducing a measure on the web and restricting to *locally bounded* morphisms, one obtains a subcategory which is a (refined) model of LLL.

Murawski and Ong were the first to obtain a full completeness result for a polytime linear logic [36]. Their model is a game semantic interpretation of the multiplicative fragment of intuitionistic light affine logic (IMLAL) and is an important contribution to the semantic analysis of the exponential connectives. The authors begin with AJM games together with the usual definition of the game $!A$ as an infinite symmetrized tensor product of copies of A. They model the exponential game $\S A$ by a single copy of game A with moves tagged by $\star$. The main idea is a generalized notion of threads, and a protocol for organizing them into networks at each depth. They call a strategy *discreet* if it obeys the network protocol at every depth up to the depth of the game. In soft linear logic, it is the degree of a proof and not the depth that is the correct invariant. Using a different notion of depth, we successfully applied their method for

handling the exponential boxes in our game model of IMSAL (see Chapter 7). This was an important step to full completeness.

Soft linear logic: The now standard strong completeness results (see [37]) for a phase semantics interpretation of ELL, EAL and SLL was established by Dal Lago and Martini in [16]. The main result in that paper is the decidability of EAL, which is obtained by showing that it satisfies a finite model property, i.e. it is complete with respect to a *finite* phase semantics. This elegant technique was developed by Lafont in [27] to give a semantic proof of the decidability of propositional linear logic with full weakening. The authors also show that the finite model property does not hold for ELL and SLL, again following Lafont's technique. There are still some interesting open questions with regards to the decidability of various fragments of linear logic. We shall discuss this point further in Chapter 8.

In [39], we show that soft linear logic possesses a very natural semantics, making it an ideal system for investigating the semantics of polynomial time computation. We give a categorical interpretation of soft linear logic and show that the exponential operator can be interpreted canonically as a certain type of limit. We call these categories *multiplexor categories.* Moreover, we give a categorical technique for constructing a multiplexor category from a finitary one which may be used if the above limit does not exist, leading to more refined models. These ideas are illustrated with concrete examples using AJM games, for example.

We also mention the unpublished work of Abramsky [2], which we were unaware of in the first draft of our paper [39]. In his Clifford lectures at Tulane University, Abramsky introduces a system of combinatory logic called Predicative Combinatory Logic (PCL), which corresponds to the $(\multimap, !)$-fragment of soft affine logic. Abramsky proves that PCL characterizes PTIME and proposes the same limit interpretation of the exponential operator, noting that such an exponential would not satisfy contraction. He also observes that the standard game models have sufficient limits for this formula to be applied. Moreover, he states that a main aim would be a full completeness theorem, thereby anticipating much of our work.

Finally, we mention the relational models for multiplicative SLL and ELL introduced by Laurent and Tortora de Falco in [30]. The main result in that paper is a *relative completeness* theorem for these models. It turns out that their model of MSLL can be obtained by applying our construction in [39]. However, both works were done completely independently [2].

Uniformly Polynomial Circuits

To motivate our ideas, we must first review some more complexity theory (we'll use [38] and [40] as references). An interesting point of view of complexity is based on *Boolean circuits*. An n-ary function $f : \{0,1\}^n \to \{0,1\}$ can be represented by a Boolean circuit with n input variables. In order to represent arbitrary languages $L \subseteq \{0,1\}^*$, we need a circuit for each possible length of the input string.

Definition 1.1.1. A *circuit family* is an infinite sequence $\mathcal{C} = (C_0, C_1, \dots)$ of Boolean circuits, where C_n has n input variables. We say that $\mathcal{C}$ decides a language $L \subseteq \{0,1\}^*$ if, for every string x, $x \in L$ if and only if $C_n(x) = 1$, where n is the length of x.

The *size* of a circuit is the number of gates that it contains. We say a circuit family $\mathcal{C} = (C_0, C_1, \dots)$ has *polynomial size complexity* if there is a fixed polynomial p such that the size of C_n is at most $p(n)$. Moreover, we need a notion of *uniformity* on circuit families which intuitively means that all circuits in the family represent the same algorithm. A circuit family $\mathcal{C} = (C_0, C_1, \dots)$ is said to be *uniform* if there is a $\log n$-space Turing machine M which on input 1^n outputs C_n. We call a uniform family of polynomial size complexity circuits a *uniformly polynomial circuit family.* Our interest derives from the following result.

Theorem 1.1.2. *A language L can be decided by a uniformly polynomial circuit family if and only if $L \in P$.*

Hence uniformly polynomial circuits can be used to study polynomial time computation. Indeed, this is a promising approach to proving that P $\neq$ NP, whereby we

[2] In fact, both results were presented at GEOCAL'06 in Marseille one after the other.

attempt to show that some specific language in NP has more than polynomial circuit complexity [40].

An important measure on soft linear logic proofs is the *rank*. We shall see that a proof net of rank n is analogous to a Boolean circuit with n input variables [3]. This idea led us to introduce a stratified interpretation of soft linear logic which is motivated by the above characterization of P.

1.2 Our Contribution

One of the goals of this thesis is to provide a mathematical perspective on polynomial time computation. As we shall see, soft linear logic is an ideal system for this purpose because it possesses a very natural mathematical interpretation. Moreover, there is a pleasing analogy to uniformly polynomial circuits that seems to be a characteristic feature of soft linear logic.

In Chapter 4, we introduce the notion of a *multiplexor category*, which is the categorical interpretation of soft linear logic. We shall see that a multiplexor category provides a denotational semantics for soft linear logic. Moreover, we shall show that the exponential operator can be interpreted canonically as a certain type of limit, giving us a *positive* reason for omitting contraction (and digging) in our models. This interpretation will motivate us to introduce a new class of AJM games in Chapter 7 which we shall call *noetherian*. Such games may be infinite, but do not have infinite length plays (i.e. there are no infinite increasing chains of plays in their game trees), hence it will be easy to show that total strategies in such games compose. In this chapter, we also show why this interpretation will not suffice to prove a full completeness result.

Motivated by notions in complexity theory, we are led to introduce a finitary version of soft linear logic and a corresponding notion of finitary multiplexor category in Chapter 5. The main result here is a categorical method for constructing a multiplexor category from a finitary one, called the S-construction. This naturally leads to

[3] A similar idea is found in [42]. It would be interesting to relate this work to ours.

a stratified view of soft linear logic which intuitively corresponds to the circuit families of the previous section. We also discuss possible categorical analogs of uniformity and polynomial complexity. This leads to a variant of the S-construction which can be used when the base category is concrete, in some sense. The latter is illustrated by giving an alternative description of a relational model of MSLL that has recently appeared in the literature [30].

In Chapter 6, we use the S-construction to give a realizability semantics for ISAL_2 and a new proof that any algorithm representable in that system is polytime, using a method pioneered by Hofmann [24]. Here we enforce polynomial size restrictions on sequences of affine lambda terms in the same way it is done for circuit families. Uniformity is built in since the underlying function at each level is the same. In other words, we are using the variant of the S-construction in which the base category is concrete, in the set-theoretic sense.

Another interesting application of the S-construction is given in Chapter 7 using a base category of *finite* AJM games. Uniformity is enforced by requiring strategies at different levels to be approximations of the same strategy, leading to a type of relative completeness result for IMSAL. Following ideas from [36], we restrict to networked strategies, and prove that our model is fully-complete for IMSAL. We also note that polynomial size and/or depth restrictions can be placed on sequences of strategies by restricting e.g. the length of plays at level n to be at most $p(n)$, for some fixed polynomial p. Again, this is consistent with our circuit analogy.

In Chapter 8, we conclude with some future directions for research.

Chapter 2

Background

This chapter collects some basic background material. Its primary purpose is for ease of reference and to fix notation. More exhaustive treatments can be found in the references. The reader familiar with this material may skip this chapter without loss of continuity and refer back to it when necessary.

2.1 Intuitionistic Linear Logic

Linear logic was introduced by J.-Y. Girard in 1987 [17] as a logical system for managing resources. This makes it ideal for investigating computation from the point of view of logic. Since our primary concern is with computation, the second-order intuitionistic system will be more than adequate for our purposes.

Formulae are given by the following syntax:

$$A, B ::= \alpha | \mathbf{1} | A \,\&\, B | A \otimes B | A \multimap B | \forall \alpha.A | !A$$

The other intuitionistic linear connectives and quantifiers are definable as follows: $A \oplus B = \forall \alpha.(A \multimap \alpha) \,\&\, (B \multimap \alpha) \multimap \alpha$, $\mathbf{0} = \forall \alpha.\alpha$, $\exists \alpha.A = \forall \beta.(\forall \alpha.A \multimap \beta) \multimap \beta$ and $\top = \exists \alpha.\alpha$ [28]. Sequents have the form $\Gamma \vdash C$, where Γ is a finite (possibly empty) list of formulas and C is a single formula. If Γ is the sequence of formulas $C_1, \ldots, C_n$, then $!\Gamma$ denotes the sequence $!C_1, \ldots, !C_n$. Proofs are generated by a Gentzen style sequent calculus, as follows.

- Structural rules: *exchange*, *identity*, and *cut*

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\ \text{ex} \qquad \frac{}{A \vdash A}\ \text{id} \qquad \frac{\Gamma \vdash A \quad \Delta, A \vdash C}{\Gamma, \Delta \vdash C}\ \text{cut}$$

- Multiplicative logical rules:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ \multimap\text{R} \qquad \frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C}\ \multimap\text{L}$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ \otimes\text{R} \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\ \otimes\text{L} \qquad \frac{}{\vdash \mathbf{1}}\ \mathbf{1}\text{R} \qquad \frac{\Gamma \vdash C}{\Gamma, \mathbf{1} \vdash C}\ \mathbf{1}\text{L}$$

- Additive logical rules:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\ \&\text{R} \qquad \frac{\Gamma, A \vdash C}{\Gamma, A \,\&\, B \vdash C}\ \&\text{L1} \qquad \frac{\Gamma, B \vdash C}{\Gamma, A \,\&\, B \vdash C}\ \&\text{L2}$$

- Quantification logical rules:

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall \alpha.A}\ \forall\text{R}\ \ (\alpha \notin \mathrm{FV}(\Gamma)) \qquad \frac{\Gamma, A[B/\alpha] \vdash C}{\Gamma, \forall \alpha.A \vdash C}\ \forall\text{L}$$

- Exponential logical rules:

$$\frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\ \text{p} \qquad \frac{\Gamma, A \vdash C}{\Gamma, !A \vdash C}\ \text{der} \qquad \frac{\Gamma, !A, !A \vdash C}{\Gamma, !A \vdash C}\ \text{c} \qquad \frac{\Gamma \vdash C}{\Gamma, !A \vdash C}\ \text{wk}$$

where the four exponential rules are, respectively, called *promotion*, *dereliction*, *contraction* and *weakening*. Note that the promotion rule may be replaced by *soft promotion* and *digging*:

$$\frac{\Gamma \vdash A}{!\Gamma \vdash !A}\ \text{sp} \qquad \frac{\Gamma, !!A \vdash C}{\Gamma, !A \vdash C}\ \text{dig}$$

Theorem 2.1.1. (Girard). ILL_2 *satisfies cut elimination.*

There is a well-known translation of second-order intuitionistic logic into ILL_2 defined as follows: $(A \Rightarrow B)^* = !A^* \multimap B^*$, $(A \wedge B)^* = A^* \,\&\, B^*$, and $(\forall \alpha.A)^* = \forall \alpha.A^*$. There is also a translation in the opposite direction defined as follows: $(A \multimap B)_* = A_* \Rightarrow B_*$, $(A \otimes B)_* = (A \,\&\, B)_* = A_* \wedge B_*$, $(!A)_* = A_*$ and $(\forall \alpha.A)_* = \forall \alpha.A_*$. In particular, this means that ILL_2 is very expressive.

Affine logic is ILL_2 together with the unrestricted rule of weakening:

$$\frac{\Gamma \vdash C}{\Gamma, A \vdash C}\ \text{wk}$$

2.2 The Light Exponentials

In this section, we shall briefly look at other linear logic characterizations of bounded time complexity.

Light linear logic was introduced by Girard in [18] as a logical system complete for polytime computation. Asperti and Roversi [5] later showed that the intuitionistic affine variant has the same good complexity properties, but has the advantage of a much cleaner syntax. Moreover, with full weakening, the computational behaviour of the additives can be simulated. For this reason, we shall only consider second-order intuitionistic multiplicative light affine logic (IMLAL_2). IMLAL_2 is IMAL_2 with exponential modalities ! (bang) and § (neutral) and exponential rules as follows:

$$\frac{A \vdash B}{!A \vdash !B} \qquad \frac{\vdash A}{\vdash !A} \qquad \frac{\Delta, \Gamma \vdash A}{!\Delta, \S\Gamma \vdash \S A} \qquad \frac{\Gamma, !A, !A \vdash C}{\Gamma, !A \vdash C}$$

This system satisfies cut elimination. Moreover, we have the following results:

Theorem 2.2.1. (Girard, Asperti).

1. *Cut elimination in a proof π can be done in time $O(s^{2^d})$, where d is the depth, and s is the size of π.*

2. *All PTIME algorithms can be encoded in $IMLAL_2$.*

Now since binary strings are represented by depth 1 proof nets, and depth is preserved under cut elimination in this system, the computation will run in polynomial time in the size of the input.

Second-order intuitionistic elementary linear logic (IELL_2) is a related system that characterizes elementary recursive computation [18]. (Elementary functions are precisely those functions computable in time bounded by a tower of exponentials $2^{2^{\cdot^{\cdot^{2^n}}}}$.) IELL_2 is ILL_2 without the exponential logical rules of dereliction and digging. So the IELL_2 exponential rules are the following:

$$\frac{\Gamma \vdash A}{!\Gamma \vdash !A} \qquad \frac{\Gamma \vdash C}{\Gamma, !A \vdash C} \qquad \frac{\Gamma, !A, !A \vdash C}{\Gamma, !A \vdash C}$$

Bounded linear logic [20], the first linear logic characterization of polynomial time computation, is based on the idea of replacing $!A$ by a bounded version $!_n A$, and bounded versions of the exponential rules. See [20] for details.

2.3 Category Theory

In this section, we briefly review some basic category theory. Excellent references include [33] and [11]. For an introduction with an emphasis on logic, see also [21], [29] and [10]. Here, we mostly follow the presentation in [33].

A *category* consists of a class of objects O and a class of arrows A, together with functions:

$$A \underset{\text{cod}}{\overset{\text{dom}}{\rightrightarrows}} O$$

which assign to each arrow a domain and a codomain. (We write $a \xrightarrow{f} b$ or $f : a \to b$ for $f \in A, \text{dom}(f) = a$ and $\text{cod}(f) = b$.) Furthermore, there is a function which assigns to each object a, an *identity* arrow $\text{id}_a : a \to a$, and a function $\circ$ which assigns to each pair of arrows of the form $f : a \to b$ and $g : b \to c$, a composite arrow $g \circ f : a \to c$ such that the following hold:

1. for all $f : a \to b$, $f \circ \text{id}_a = f = \text{id}_b \circ f$
2. for all $a \xrightarrow{f} b \xrightarrow{g} c \xrightarrow{k} d$, $k \circ (g \circ f) = (k \circ g) \circ f$.

Given two objects a and b in a category C, we write $\text{hom}_C(a, b)$ or sometimes $C(a, b)$ for the set of arrows $f : a \to b$ in C.

A simple example of a category is the category **Rel** of sets and relations. An object is any small set, and an arrow $R : X \to Y$ is any binary relation $R \subseteq X \times Y$. Given two relations $R \subseteq X \times Y$ and $S \subseteq Y \times Z$, the composite relation $S \circ R$ is defined as follows:

$$S \circ R = \{\langle x, z\rangle |\ \exists y \in Y, \langle x, y\rangle \in R \text{ and } \langle y, z\rangle \in S\}$$

The identity arrow $\text{id}_X : X \to X$ is defined as the relation $\{\langle x, x\rangle |\ x \in X\} \subseteq X \times X$. It is easy to check that this satisfies the axioms of a category.

A *functor* T between two categories C and B consists of a pair of functions which assign to each object $c \in C$, an object $T(c) \in B$, and to each arrow $f : c \to d$ in C, an arrow $T(f) : T(c) \to T(d)$ in B, in such a way that

$$T(\text{id}_c) = \text{id}_{T(c)} \qquad T(g \circ f) = T(g) \circ T(f)$$

A functor $T : C \to B$ is *full* when to every pair of objects c and d in C, and to every arrow $g : T(c) \to T(d)$ in B, there is an arrow $f : c \to d$ in C such that $g = T(f)$. A functor $T : C \to B$ is *faithful* (or an embedding) when to every pair of objects $c, d \in C$ and to every pair of parallel arrows $f_1, f_2 : c \to d$ in C, the equality $T(f_1) = T(f_2) : T(c) \to T(d)$ implies $f_1 = f_2$.

A *subcategory* S of a category C consists of some of the objects of C and some of the arrows of C such that 1) for each arrow f in S, both $\mathrm{dom}(f)$ and $\mathrm{cod}(f)$ are objects in S, 2) for each object s in S, its identity arrow id_s is an arrow in S, and 3) for each pair of composable arrows $s \xrightarrow{f} s' \xrightarrow{g} s''$ in S, the composite arrow $g \circ f : s \to s''$ is in S. We say that S is a *full subcategory* of C when the inclusion functor $S \to C$ is full. For example, the category $\mathbf{Rel}_f$ of *finite* sets and relations is a full subcategory of **Rel**.

Natural Transformations

Given two functors $S, T : C \to B$, a *natural transformation* $\alpha : S \dot{\to} T$ is a function which assigns to each object c in C an arrow $\alpha_c : S(c) \to T(c)$ in B in such a way that for all $f : c \to c'$ in C, the following diagram commutes in B:

$$\begin{array}{ccc} S(c) & \xrightarrow{\alpha_c} & T(c) \\ {\scriptstyle S(f)}\downarrow & & \downarrow{\scriptstyle T(f)} \\ S(c') & \xrightarrow{\alpha_{c'}} & T(c') \end{array}$$

In this case, we say that $\alpha_c : S(c) \to T(c)$ is *natural* in c, and we call the arrows $\alpha_a, \alpha_b, \alpha_c, \ldots$, the *components* of the natural transformation α. Let $\alpha : S \dot{\to} T$ be a natural transformation between functors $S, T : C \to B$. If every component α_c is invertible in B, then there is an inverse natural transformation $\alpha^{-1} : T \dot{\to} S$ defined by $\alpha^{-1}(c) = \alpha(c)^{-1}$. In this case, we call α a *natural isomorphism* and we write $S \cong T$. Two categories C and B are called *equivalent* if there is a pair of functors $S : C \to B$ and $T : B \to C$ such that $TS \cong I : C \to C$ and $ST \cong I : B \to B$. In this case, the functor S (and the functor T) is called an *equivalence of categories.*

Adjunctions

The following concept plays a central role in category theory.

Definition 2.3.1. [33] Let A and X be categories. An adjunction from X to A is a triple $\langle F, G, \varphi \rangle : X \to A$, where F and G are functors

$$X \underset{G}{\overset{F}{\rightleftarrows}} A$$

and φ is a function which assigns to each pair of objects $x \in X$, $a \in A$ a bijection of sets

$$\varphi = \varphi_{x,a} : A(Fx, a) \cong X(x, Ga) \tag{1}$$

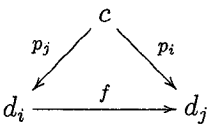
which is natural in x and a. In this case we say that F is a *left-adjoint* for G and G is a *right-adjoint* for F.

For each object $x \in X$, we can instantiate the a in the left hand side of (1) with Fx and look at the image under φ of the identity id_{Fx}. This yields an arrow $\eta_x : x \to GFx$ in X which is the component of a natural transformation $\eta : I_X \dot{\to} GF$, called the *unit* of the adjunction. Similarly, for each object $a \in A$, we can instantiate the x in the right hand side of (1) with Ga, and look at the image under φ^{-1} of id_{Ga}. This yields an arrow $\varepsilon_a : FGa \to a$ in A which is the component of a natural transformation $\varepsilon : FG \dot{\to} I_A$, called the *counit* of the adjunction. Moreover, for any pair of objects $x \in X, a \in A$, we have $G(\varepsilon_a) \circ \eta_{Ga} = \mathrm{id}_{Ga}$ in X and $\varepsilon_{Fx} \circ F(\eta_x) = \mathrm{id}_{Fx}$ in A. In fact, the unit and counit completely determine the adjunction. It is now easy to check that if $S : C \to B$ is an equivalence of categories with backwards functor $T : B \to C$, then T is both a left and a right adjoint of S.

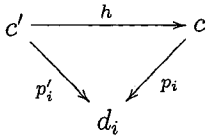
Limits

A *diagram* D in a category C is simply a collection of some C-objects $d_i, d_j, \ldots$ together with some C-arrows $f : d_i \to d_j$ between those objects. A finite diagram is one that has a finite number of objects, and a finite number of arrows between them. A *cone* for a diagram D, or simply a *D-cone*, consists of a C-object c together with

a C-arrow $p_i : c \to d_i$ for each object d_i of D, such that



commutes, whenever f is an arrow in the diagram D. A D-cone is denoted by $\{p_i : c \to d_i\}$. A *limit* for a diagram D is a *universal* D-cone. In other words, a limit is a D-cone $\{p_i : c \to d_i\}$ with the property that for any other D-cone $\{p'_i : c' \to d_i\}$, there is a unique C-arrow $h : c' \to c$ such that



commutes, for every object d_i in diagram D. It is easy to show that the limit c of a diagram D, when it exists, is unique up to isomorphism.

For example,

1. A limit of the empty diagram is called a *terminal* object.

2. Given C-objects a and b, a limit of the diagram

   $$a \qquad b$$

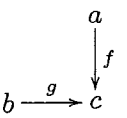
   is called a *binary product* of a and b.

3. Given C-objects a and b and parallel C-arrows $f, g : a \to b$, a limit of the diagram

   $$a \underset{g}{\overset{f}{\rightrightarrows}} b$$

   is called an *equalizer* of f and g.

4. Given C-objects a, b and c and C-arrows $f : a \to c$ and $g : b \to c$, a limit of the diagram



is called a *pullback* of f and g.

A category C is *(finitely) complete* if every (finite) diagram in C has a limit in C. The dual notion of *colimit* will not be discussed here.

The following is a useful proposition.

Proposition 2.3.2. *Let $\{p_i : c \to d_i\}$ be a limit of a diagram D in a category C. Two parallel C-arrows $f, g : a \to b$ are equal iff for all objects $d_i \in D$, $p_i \circ f = p_i \circ g$.*

Monoidal Categories

A *monoidal* category $B = \langle B, \otimes, e, \alpha, \lambda, \rho \rangle$ is a category B, a bifunctor $\otimes : B \times B \to B$, an object $e \in B$, and three natural isomorphisms:

$$\begin{aligned} \alpha &= \alpha_{a,b,c} : a \otimes (b \otimes c) \cong (a \otimes b) \otimes c \\ \lambda &= \lambda_a : e \otimes a \cong a \\ \rho &= \rho_a : a \otimes e \cong a \end{aligned}$$

natural for all $a, b, c \in B$. These isomorphisms are required to satisfy the following diagrams:

$$\begin{array}{ccccc} a \otimes (b \otimes (c \otimes d)) & \xrightarrow{\alpha} & (a \otimes b) \otimes (c \otimes d) & \xrightarrow{\alpha} & ((a \otimes b) \otimes c) \otimes d \\ \downarrow{\scriptstyle \mathrm{id}\otimes\alpha} & & & & \uparrow{\scriptstyle \alpha\otimes\mathrm{id}} \\ a \otimes ((b \otimes c) \otimes d) & & \xrightarrow{\qquad\qquad\alpha\qquad\qquad} & & (a \otimes (b \otimes c)) \otimes d \end{array}$$

$$\begin{array}{ccc} a \otimes (e \otimes c) & \xrightarrow{\alpha} & (a \otimes e) \otimes c \\ & {\scriptstyle \mathrm{id}\otimes\lambda} \searrow \quad \swarrow {\scriptstyle \rho\otimes\mathrm{id}} & \\ & a \otimes c & \end{array}$$

$$\lambda_e = \rho_e : e \otimes e \to e$$

Note that $\otimes$ a bifunctor means that the interchange law holds

$$\mathrm{id}_a \otimes \mathrm{id}_b = \mathrm{id}_{a\otimes b}, \qquad (f' \otimes g') \circ (f \otimes g) = (f' \circ f) \otimes (g' \circ g),$$

whenever the composites $f' \circ f$ and $g' \circ g$ are defined.

A monoidal category is called *strict* when the structure maps α, λ and ρ are all identities.

Coherence

Let $\langle B, \otimes, e_0, \dots \rangle$ be a monoidal category. A *binary word* is given by the following syntax:

$$v, w ::= e_0| - |(v \otimes w)$$

The *length* lh of a binary word is defined inductively as follows:

$$\begin{aligned} lh(e_0) &= 0 \\ lh(-) &= 1 \\ lh(v \otimes w) &= lh(v) + lh(w) \end{aligned}$$

Each binary word w of length n determines an n-ary functor $w_B : B^n \to B$ in the obvious way. Then the coherence theorem takes the following form:

Theorem 2.3.3. (Mac Lane). *Let B be a monoidal category. There is a function which assigns to each pair of words v, w of the same length n a (unique) natural isomorphism*

$$\mathrm{can}_B(v, w) : v_B \dot{\to} w_B : B^n \to B$$

called the canonical map from v_B to w_B, in such a way that the identity arrow $e \to e$ is canonical (between functors of 0 variables), the identity transformation $\mathrm{id}_B : I_B \dot{\to} I_B$ is canonical, α, α^{-1}, λ, λ^{-1}, ρ and ρ^{-1} are canonical, and the composite as well as the $\otimes$-product of two canonical maps is canonical.

This theorem states that every diagram of the following form commutes:

- *Vertices.* Words w of length n representing functors $w_B : B^n \to B$.
- *Edges.* Natural transformations 1_e, id_B,α, λ, ρ and their $\otimes$ products.

Symmetric Monoidal Closed Categories

A *symmetric monoidal category* is a monoidal category $B = \langle B, \otimes, e, \alpha, \lambda, \rho \rangle$ equipped with an additional isomorphism:

$$\gamma = \gamma_{a,b} : a \otimes b \cong b \otimes a,$$

natural in $a, b \in B$. This natural isomorphism is required to satisfy the following diagrams:

$$\gamma_{a,b} \circ \gamma_{b,a} = \mathrm{id}, \qquad \rho_b = \lambda_b \circ \gamma_{b,e} : b \otimes e \cong b,$$

$$\begin{array}{ccccc} a \otimes (b \otimes c) & \xrightarrow{\alpha} & (a \otimes b) \otimes c & \xrightarrow{\gamma} & c \otimes (a \otimes b) \\ \downarrow{\scriptstyle \mathrm{id} \otimes \gamma} & & & & \downarrow{\scriptstyle \alpha} \\ a \otimes (c \otimes b) & \xrightarrow{\alpha} & (a \otimes c) \otimes b & \xrightarrow{\gamma \otimes \mathrm{id}} & (c \otimes a) \otimes b \end{array}$$

The coherence theorem for monoidal categories can be generalized to the symmetric case as follows. Let B be a symmetric monoidal category. A *permuted word* $w\tau$ consists of a word w together with a permutation $\tau \in S_n$. This determines a functor $(w\tau)_B : B^n \to B$ defined by $(w\tau)_B(a_1, \ldots, a_n) = w(a_{\tau 1}, \ldots, a_{\tau n})$, $a_i \in B$.

Theorem 2.3.4. (Mac Lane). *Let B be a symmetric monoidal category. There is a function which assigns to each pair of permuted words $(v\sigma, w\tau)$ of the same length n a (unique) natural isomorphism*

$$\mathrm{can}_B(v\sigma, w\tau) : (v\sigma)_B \dot{\to} (w\tau)_B : B^n \to B$$

called the canonical map from $v\sigma$ to $w\tau$, in such a way that the identity arrow in B and all instances of α, λ, ρ and γ are canonical, and the composite as well as the $\otimes$-product of two canonical maps is canonical.

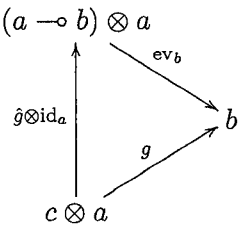
This theorem means that all formal diagrams involving just α, λ, ρ and γ will commute.

A *symmetric monoidal closed category* B is a symmetric monoidal category in which each functor $- \otimes a : B \to B$ has a specified right adjoint $a \multimap - : B \to B$. Equivalently, for any two objects $a, b \in B$, there is a B-object $a \multimap b$ together with a B-arrow

$$\mathrm{ev}_b : (a \multimap b) \otimes a \to b$$

called *evaluation* with the following universal property. For any B-object c and B-arrow $g : c \otimes a \to b$, there is a unique B-arrow $\hat{g} : c \to a \multimap b$ making the following

diagram commute:



The assignment $g \mapsto \hat{g}$ establishes the natural bijection:

$$\hom_B(a \otimes b, c) \cong \hom_B(a, b \multimap c)$$

The natural transformation ev turns out to be the counit of the adjunction. The unit coev has components:

$$\mathrm{coev}_x : x \longrightarrow a \multimap (x \otimes a)$$

These satisfy the usual adjunction equations.

Monoidal Functors

A *symmetric monoidal functor* $(F, F_2, F_0) : M \to M'$ between symmetric monoidal categories $M = \langle M, \otimes, e, \alpha, \lambda, \rho, \gamma \rangle$ and $M' = \langle M', \otimes, e', \alpha', \lambda', \rho', \gamma' \rangle$ consists of the following three items:

1. An ordinary functor $F : M \to M'$ between categories;

2. For any objects $a, b \in M$, an arrow

$$F_2(a, b) : F(a) \otimes F(b) \to F(a \otimes b)$$

   in M' which is natural in a and b;

3. For the units $e \in M$ and $e' \in M'$, an arrow in M'

$$F_0 : e' \to F(e)$$

These are required to make the following four diagrams commute:

$$\begin{array}{ccc} F(a)\otimes(F(b)\otimes F(c)) & \xrightarrow{\alpha'} & (F(a)\otimes F(b))\otimes F(c) \\ \downarrow{\scriptstyle \mathrm{id}_{Fa}\otimes F_2} & & \downarrow{\scriptstyle F_2\otimes \mathrm{id}_{Fc}} \\ F(a)\otimes F(b\otimes c) & & F(a\otimes b)\otimes F(c) \\ \downarrow{\scriptstyle F_2} & & \downarrow{\scriptstyle F_2} \\ F(a\otimes(b\otimes c)) & \xrightarrow{F(\alpha)} & F((a\otimes b)\otimes c) \end{array} \tag{2}$$

$$\begin{array}{ccc} F(b)\otimes e' & \xrightarrow{\rho'} & F(b) \\ \downarrow{\scriptstyle \mathrm{id}_{Fb}\otimes F_0} & & \uparrow{\scriptstyle F(\rho)} \\ F(b)\otimes F(e) & \xrightarrow{F_2} & F(b\otimes e) \end{array} \qquad \begin{array}{ccc} e'\otimes F(b) & \xrightarrow{\lambda'} & F(b) \\ \downarrow{\scriptstyle F_0\otimes \mathrm{id}_{Fb}} & & \uparrow{\scriptstyle F(\lambda)} \\ F(e)\otimes F(b) & \xrightarrow{F_2} & F(e\otimes b) \end{array} \tag{3}$$

$$\begin{array}{ccc} F(a)\otimes F(b) & \xrightarrow{\gamma'} & F(b)\otimes F(a) \\ \downarrow{\scriptstyle F_2} & & \downarrow{\scriptstyle F_2} \\ F(a\otimes b) & \xrightarrow{F(\gamma)} & F(b\otimes a) \end{array} \tag{4}$$

Observe that the composite of two symmetric monoidal functors is a symmetric monoidal functor. A monoidal functor is called *strong* when F_0 and all $F_2(a,b)$ are isomophisms, and *strict* when F_0 and all $F_2(a,b)$ are identities.

A *monoidal natural transformation* $\theta : (F, F_2, F_0) \dot{\to} (G, G_2, G_0) : M \to M'$ between two monoidal functors is an ordinary natural transformation $\theta : F \dot{\to} G$ such that all the diagrams

$$\begin{array}{ccc} F(a)\otimes F(b) & \xrightarrow{F_2} & F(a\otimes b) \\ \downarrow{\scriptstyle \theta_a\otimes\theta_b} & & \downarrow{\scriptstyle \theta_{a\otimes b}} \\ G(a)\otimes G(b) & \xrightarrow{G_2} & G(a\otimes b) \end{array} \qquad \begin{array}{ccc} e' & \xrightarrow{F_0} & F(e) \\ \| & & \downarrow{\scriptstyle \theta_e} \\ e' & \xrightarrow{G_0} & G(e) \end{array} \tag{5}$$

commute in M'. The compose of two monoidal natural transformations is a monoidal natural transformation.

Let $(F, F_2, F_0) : M \to M'$ be a symmetric monoidal functor and $v\sigma$ a permuted word of length n. There is an arrow

$$F_{v\sigma} : v\sigma(Fa_1, \ldots, Fa_n) \to Fv\sigma(a_1, \ldots, a_n)$$

in M' natural in $a_1, \ldots, a_n$. Then for any two permuted words $v\sigma, w\tau$ of length n, it follows that the following diagram

$$\begin{array}{ccc} v\sigma(Fa_1, \ldots, Fa_n) & \xrightarrow{F_{v\sigma}} & Fv\sigma(a_1, \ldots, a_n) \\ \downarrow{\scriptstyle \eta'} & & \downarrow{\scriptstyle F\eta} \\ w\tau(Fa_1, \ldots, Fa_n) & \xrightarrow{F_{w\tau}} & Fw\tau(a_1, \ldots, a_n) \end{array}$$

commutes in M', where η' and η are given uniquely by the coherence theorem for symmetric monoidal categories. Moreover, for any monoidal natural transformation $\theta : F \dot{\rightarrow} G$ between two symmetric monoidal functors and for any permuted word $v\sigma$ of length n, the diagram

$$\begin{array}{ccc} v\sigma(Fa_1, \ldots, Fa_n) & \xrightarrow{F_{v\sigma}} & Fv\sigma(a_1, \ldots, a_n) \\ \downarrow{\scriptstyle v\sigma(\theta_{a_1}, \ldots, \theta_{a_n})} & & \downarrow{\scriptstyle \theta_{v\sigma(\theta_{a_1}, \ldots, \theta_{a_n})}} \\ v\sigma(Ga_1, \ldots, Ga_n) & \xrightarrow{G_{v\sigma}} & Gv\sigma(a_1, \ldots, a_n) \end{array}$$

commutes in M'. This generalizes the six diagrams in the definition of symmetric monoidal functor and monoidal natural transformation.

Comonoids

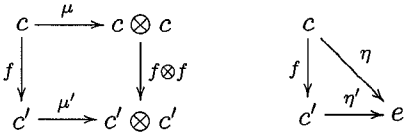
A *commutative comonoid* c in a symmetric monoidal category $B = \langle B, \otimes, e, \alpha, \lambda, \rho, \gamma \rangle$ consists of an object $c \in B$ together with two arrows $\mu : c \to c \otimes c$ and $\eta : c \to e$ such that the diagrams

$$\begin{array}{ccccc} c & & \xrightarrow{\mu} & & c \otimes c \\ \downarrow{\scriptstyle \mu} & & & & \downarrow{\scriptstyle \mu \otimes \mathrm{id}_c} \\ c \otimes c & \xrightarrow{\mathrm{id}_c \otimes \mu} & c \otimes (c \otimes c) & \xrightarrow{\alpha} & (c \otimes c) \otimes c \end{array}$$

$$\begin{array}{ccccc} e \otimes c & \xrightarrow{\lambda} & c & \xleftarrow{\rho} & c \otimes e \\ & {\scriptstyle \eta \otimes \mathrm{id}_c} \searrow & \downarrow{\scriptstyle \mu} & \swarrow {\scriptstyle \mathrm{id}_c \otimes \eta} & \\ & & c \otimes c & & \end{array}$$

$$\begin{array}{ccc} c & \xrightarrow{\mu} & c \otimes c \\ & {\scriptstyle \mu} \searrow & \downarrow{\scriptstyle \gamma} \\ & & c \otimes c \end{array}$$

commute in B. A morphism $f : \langle c, \mu, \eta \rangle \to \langle c', \mu', \eta' \rangle$ of comonoids is an arrow $f : c \to c'$ in B such that the diagrams



commute in B. With these arrows, the comonoids in B form a category $\mathbf{CMon}_B$, and $\langle c, \mu, \eta \rangle \mapsto c$ defines a forgetful functor $U : \mathbf{CMon}_B \to B$.

Chapter 3

Soft Linear Logic and PCL

In 2001, Y. Lafont introduced a subsystem of second-order intuitionistic linear logic, called soft linear logic (ISLL_2), with restricted rules for exponentials [28]. Lafont shows that proofs in ISLL_2 correspond to polynomial time algorithms, and vice versa. In other words, ISLL_2 is complete for polynomial time computation. The key point is to remove contraction $!A \multimap !A \otimes !A$ and digging $!A \multimap !!A$ and replace them with a new rule called *multiplexing*, which corresponds to the axiom:

$$!A \multimap A^n \qquad (n \text{ is any natural number})$$

where A^n denotes the n-fold tensor product. To my knowledge, all other linear logic characterizations of bounded-time computation use contraction in one form or another. This highlights a significant difference between the various approaches.

In a parallel work, S. Abramsky introduced a system called Predicative Combinatory Logic, or PCL, in his Clifford lectures at Tulane University [2]. As noted by Abramsky, the computational power of copying is clearly seen in systems of combinatory logic. For example, in **BCKW** logic, all partial recursive functions are representable; however, if we remove the contraction combinator **W**, the resulting system has the property that every term reduces to normal form in linear time. In PCL, the **W** combinator is replaced by a family of combinators:

$$\{\mathbf{W}^n |\ n > 0\}$$

with defining equations:

$$\mathbf{W}^n x!y = x\underbrace{y\cdots y}_{n}$$

The intuition is best described by Abramsky when he compares $!t$ to a promissory note, or a "blank cheque" which gets consumed or cashed in by applying $\mathbf{W}^n$. Peter Selinger (private communication) as well as myself have also independently used this analogy [39]. Abramsky uses the term *predicative copying* to describe multiplexing, to contrast the recursivity, or unboundedness, or impredicativity of contraction [2], hence the name Predicative Combinatory Logic. Abramsky shows that PCL characterizes PTIME in a similar way to Lafont's characterization. In fact, when we consider type inferencing of PCL terms, we shall see that PCL corresponds to the $(\multimap, !)$-fragment of soft affine logic.

3.1 Soft Linear Logic

In this section we review the theory of second-order intuitionistic soft linear logic (ISLL_2). Formulae are given by the following syntax:

$$A, B ::= \alpha | \mathbf{1} | A \,\&\, B | A \otimes B | A \multimap B | \forall\alpha.A | !A$$

The other intuitionistic linear connectives and quantifiers are definable as follows: $A \oplus B = \forall\alpha.(A \multimap \alpha) \,\&\, (B \multimap \alpha) \multimap \alpha$, $\mathbf{0} = \forall\alpha.\alpha$, $\exists\alpha.A = \forall\beta.(\forall\alpha.A \multimap \beta) \multimap \beta$ and $\top = \exists\alpha.\alpha$ [28]. Sequents have the form $\Gamma \vdash C$, where Γ is a finite (possibly empty) list of formulas and C is a single formula. If Γ is the sequence of formulas $C_1, \ldots, C_n$, then $!\Gamma$ denotes the sequence $!C_1, \ldots, !C_n$. Also, if A is a formula and $n \in \mathbb{N}$, we write A^n for the formula $A \otimes \cdots \otimes A$ (n times) and $A^{(n)}$ for the sequence $A, \ldots, A$ (n times). Proofs are generated by a Gentzen style sequent calculus. The rules are the following (as originally presented in [28]):

- structural rules: *exchange*, *identity*, and *cut*

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C} \qquad \frac{}{A \vdash A} \qquad \frac{\Gamma \vdash A \quad \Delta, A \vdash C}{\Gamma, \Delta \vdash C}$$

- multiplicative logical rules:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \qquad \frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C}$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C} \qquad \frac{}{\vdash \mathbf{1}} \qquad \frac{\Gamma \vdash C}{\Gamma, \mathbf{1} \vdash C}$$

- additive logical rules:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B} \qquad \frac{\Gamma, A \vdash C}{\Gamma, A \,\&\, B \vdash C} \qquad \frac{\Gamma, B \vdash C}{\Gamma, A \,\&\, B \vdash C}$$

- exponential logical rules: *soft promotion* and *multiplexing of rank n*

$$\frac{\Gamma \vdash A}{!\Gamma \vdash !A} \qquad \frac{\Gamma, A^{(n)} \vdash C}{\Gamma, !A \vdash C} \quad (n \text{ is any natural number})$$

- quantification logical rules:

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall \alpha . A} \quad (\alpha \text{ not free in } \Gamma) \qquad \frac{\Gamma, A[B/\alpha] \vdash C}{\Gamma, \forall \alpha . A \vdash C}$$

Note that one can recover the usual Linear Logic exponentials by adding the *digging* rule:

$$\frac{\Gamma, !!A \vdash C}{\Gamma, !A \vdash C}$$

to the above set of rules. The following result is due to Lafont [28].

Theorem 3.1.1. (Lafont). ISLL_2 *satisfies cut-elimination.*

For our purposes, we found it more convenient to present a cut-elimination theorem directly from the syntax, rather than using proof nets. Therefore, we give another proof of cut-elimination in appendix A for the (quantifier-free) multiplicative fragment.

A *proof net* is a graphical representation of proofs invented by Girard which can be seen as an equivalence class of proofs and provides an alternative syntax for proofs. In what follows, we shall identify proofs with proof nets, since it simplifies some of the arguments and definitions. Proof nets for ISLL_2 can be found in [28]. Given a proof net u, the *degree*, the *rank*, and the *weight* of u are defined as follows:

- The *degree* of u counts the maximum nesting of exponential boxes in u.
- The *rank* of u is the maximal rank of multiplexors in u. If all these ranks are the same, we say that u is *homogeneous*, and if there is no multiplexing, we say that u is *generic*. A generic proof net will be considered as a homogeneous proof net of rank n for any n [28]. In ISLL_2 programming, programs are represented by generic proofs and data by bounded-degree homogeneous proofs.
- The *weight* of u is a polynomial W_u, defined inductively as follows. The weight of an axiom is constant 1 and the weight is unchanged by an exchange rule or a cut rule. The weight increases by 1 for any right logical rule except soft promotion and the weight stays the same for all left logical rules. For soft promotion, we have:

$$W_{!u} = XW_u + 1$$

We get the following normalization property:

Theorem 3.1.2. (Lafont). *A proof net u of rank n reduces to a unique normal form in at most $W_u(n)$ steps.*

Uniqueness is a consequence of the proof net formulation, which identifies proofs that are "morally" the same. For any proof net u of degree d, we have $W_u(n) \leq kn^d$ for all $n \geq 1$, where $k = W_u(1)$. So if datatypes are represented by bounded degree proof nets, we get normalization in polynomial time. In general, we get normalization in exponential time.

The key step to showing that ISLL_2 can represent all PTIME algorithms is to show how to represent polynomial expressions in ISLL_2. (A polynomial expression is just a term built from natural numbers and a variable X, using addition and multiplication.) The notation A^n is extended to polynomial expressions as follows:

$$A^X = !A \qquad A^{P+Q} = A^P \otimes A^Q \qquad A^{PQ} = (A^P)^Q$$

Lafont writes $A\langle P\rangle$ for the formula A where each subformula of the form $!B$ is replaced by B^P. For example, $!\alpha\langle XX2+1\rangle = \alpha^{XX2+1} = \alpha^{XX2} \otimes \alpha = (!!\alpha)^2 \otimes \alpha$, where α is an atom.

Theorem 3.1.3. (Lafont). *If the sequent $\Gamma \vdash A$ has a homogeneous proof of rank $P(n)$, then $\Gamma\langle P\rangle \vdash A\langle P\rangle$ has one of rank n.*

For example, the formula $A = !\alpha \multimap \alpha^8$ has a homogeneous proof of rank 8, hence $A\langle XXX\rangle = !!!\alpha \multimap \alpha^8$ has a homogeneous proof of rank 2.

It is possible to represent natural numbers, strings, booleans, and Turing machines (with k states and 3 symbols, say 0,1 and *blank*) in ISLL_2 by bounded degree nets. The corresponding types are as follows:

$$\begin{aligned}
\mathbf{N} &= \forall\alpha.!(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha \\
\mathbf{S} &= \forall\alpha.!((\alpha \multimap \alpha) \,\&\, (\alpha \multimap \alpha)) \multimap \alpha \multimap \alpha \\
\mathbf{B} &= \forall\alpha.(\alpha \,\&\, \alpha) \multimap \alpha \\
\mathbf{M} &= \forall\alpha.!((\alpha \multimap \alpha) \,\&\, (\alpha \multimap \alpha)) \multimap \mathbf{F}_k \otimes \mathbf{F}_3 \otimes (\alpha \multimap \alpha)^2
\end{aligned}$$

For each natural number n, the type $\mathbf{F}_n = \mathbf{1} \oplus \cdots \oplus \mathbf{1}$ (n times) is called a *finite type.* In the type of $\mathbf{M}$, the finite types $\mathbf{F}_k$ and $\mathbf{F}_3$ record the current state and symbol, respectively.

It is possible to build [28]:

- a generic proof net for the sequent $\mathbf{S} \vdash \mathbf{N}$ which calculates the length of a boolean string.
- a generic proof net for the sequent $\mathbf{M} \vdash \mathbf{M}$ which corresponds to the transition map of this machine.
- a generic proof net for the sequent $\mathbf{N} \vdash \mathbf{M}$ which transforms a natural number n into a machine with a tape of length n (filled with 0) and with the head at the beginning of the tape.
- a generic proof net for the sequent $\mathbf{M} \vdash \mathbf{B}$ which says if a machine is in an accepting state or non-accepting state.

Theorem 3.1.4. (Lafont). *If a predicate on boolean strings in computable by a Turing machine in polynomial time $P(n)$ and polynomial space $Q(n)$, there is a generic proof for the sequent $\mathbf{S}^{(\deg P + \deg Q + 1)} \vdash \mathbf{B}$ which corresponds to this predicate.*

Proof. (High-level description.) We assume that polynomials P and Q are represented by polynomial expressions (by using their *Horner normal form* [28], for instance). Using the generic proof net for the transition function of the machine and theorem 3.1.3, we get a generic proof net for the sequent $\mathbf{M}\langle Q\rangle \vdash \mathbf{M}\langle Q\rangle$ which corresponds to the transition map of a space-bounded Turing machine which uses at most $Q(n)$ cells on an input of size n. Then there is a generic proof net for the sequent $\mathbf{N}\langle P\rangle, \mathbf{M}\langle Q\rangle \vdash \mathbf{M}\langle Q\rangle$ which corresponds to the full computation of the machine with at most $P(n)$ transitions. We also have a generic proof net for the sequent $\mathbf{S}^{(\deg P)} \vdash \mathbf{N}\langle P\rangle$ which gives a bound on the total number of steps in the computation. Moreover, we have a generic proof net for the sequent $\mathbf{S}^{(\deg Q)} \vdash \mathbf{M}\langle Q\rangle$ which generates the machine, and one for the sequent $\mathbf{S}, \mathbf{M}\langle Q\rangle \vdash \mathbf{M}\langle Q\rangle$ which writes a string of size n on the machine's tape (assuming $n \leq Q(n)$). Finally, we have a generic proof net for the sequent $\mathbf{M}\langle Q\rangle \vdash \mathbf{B}$ which says if the machine is in an accepting state. By combining all those nets, we get a generic proof net for the sequent $\mathbf{S}^{(\deg P + \deg Q + 1)} \vdash \mathbf{B}$ which corresponds to the predicate. □

By theorem 3.1.2 any generic proof net for the sequent $\mathbf{S}^{(n)} \vdash \mathbf{B}$ defines a polynomial time algorithm that takes a boolean string as input and returns a boolean value as output. Conversely, by theorem 3.1.4, any such polynomial time algorithm is represented by a generic proof net. In other words, ISLL_2 is complete for polynomial time computation [28].

Observe that the above encoding uses the additive connectives. It was conjectured by Lafont in [28] that IMSLL_2 is also complete for polynomial time computation. Indeed, this was later proved by Mairson and Terui in [34], where an alternative encoding is used:

$$\begin{aligned} \mathbf{B} &= \forall\alpha.\alpha \multimap \alpha \multimap \alpha \otimes \alpha \\ \mathbf{S} &= \forall\alpha.!(\mathbf{B} \multimap \alpha \multimap \alpha) \multimap \alpha \multimap \alpha \\ \mathbf{M}[A] &= \forall\alpha.!(\mathbf{B} \multimap \alpha \multimap \alpha) \multimap (\alpha \multimap \alpha)^k \otimes A \end{aligned}$$

where the latter is the type of a Turing machine with k stacks. The type A plays a similar role to the finite types used in Lafont's encoding of Turing machines. In [34],

they use a machine with 5 stacks. This more complicated encoding is partly due to the fact that the authors stay true to the requirement that programs be represented by generic proofs (i.e. no multiplexing). For this reason, we felt it justified to focus mainly on the multiplicative fragment of soft linear logic in our work.

3.2 Predicative Combinatory Logic

Independent of this work, S. Abramsky introduced Predicative Combinatory Logic, or PCL, in his Clifford lectures at Tulane University [2]. PCL is an orthogonal (i.e. left-linear and non-overlapping) term-rewriting system that characterizes PTIME computation, and in fact corresponds to the $(\multimap, !)$-fragment of soft affine logic. We describe the system as follows.

We assume an infinite sequence of *variables* $x, y, z, \ldots$, possibly with subscripts. Moreover, we have an infinite number of constants, denoted $\mathbf{B}$, $\mathbf{C}$, $\mathbf{K}$, $\mathbf{F}$ and $\mathbf{W}^n$ $(n > 0)$. The set of PCL-terms is defined inductively as follows:

- All variables and constants are PCL-terms.
- If t and u are PCL-terms then so are (tu) and $(!t)$.

As usual, a *combinator* is a term containing no variables. For example, $(\mathbf{W}^2\mathbf{B})$ is a combinator. For notational convenience, we shall use the following conventions for writing terms:

1. we omit outermost parentheses; thus tu means (tu)
2. application associates to the left; thus tuv means $(tu)v$
3. the scope of a ! extends as far to the right as possible; thus $!tu$ means $!(tu)$.

We have the following reduction rules for these combinators[1]:

[1]We also assume, if $t \to t'$, then $!t \to !t'$, $st \to st'$ and $ts \to t's$, for any term s.

$$\begin{aligned}
\mathbf{B}xyz &\rightarrow x(yz) \\
\mathbf{C}xyz &\rightarrow xzy \\
\mathbf{K}xy &\rightarrow x \\
\mathbf{F}!x!y &\rightarrow !(xy) \\
\mathbf{W}^n x!y &\rightarrow x\underbrace{y\cdots y}_{n}
\end{aligned}$$

For example, $\mathbf{CKK}x \rightarrow \mathbf{K}x\mathbf{K} \rightarrow x$, hence the combinator $\mathbf{CKK}$ may serve as an identity combinator. We shall write $\rightarrow^*$ for the reflexive, transitive closure of $\rightarrow$. Observe that PCL is an orthogonal term-rewriting system, hence confluent (i.e. the Church-Rosser property holds) [2].

Normalization in PCL

In [2], Abramsky proves that PCL-terms are strongly normalizing by introducing the following measures on terms:

- The *size of a term* measures the number of leaves in the term tree:

$$s(c) = 1 \qquad s(tu) = s(t) + s(u) \qquad s(!t) = s(t)$$

- The *depth of a term* measures the maximum nesting depth of !s:

$$d(c) = 0 \qquad d(tu) = \max(d(t), d(u)) \qquad d(!t) = 1 + d(t)$$

- The *width of a term* is defined by:

$$w(t) = \max(\{n |\ \mathbf{W}^n \text{ occurs in } t\} \cup \{1\})$$

- For each term t we associate a polynomial $P_t(x)$ in the variable x:

$$P_c(x) = 1 \qquad P_{tu}(x) = P_t(x) + P_u(x) \qquad P_{!t}(x) = x \cdot P_t(x)$$

  Then the *weight of a term* is defined as: $\|t\| = P_t(w(t))$.

The following propositions are due to Abramsky [2].

Proposition 3.2.1. (Abramsky). *For all t: $\|t\| \leq w^d s$, where $w = w(t)$, $s = s(t)$ and $d = d(t)$. Thus the weight is linear in the size, polynomial in the width, and exponential in the depth.*

Proof. Note that $\|t\| = P_t(w(t)) \leq w^d \cdot P_t(1) = w^d \cdot s$. □

Proposition 3.2.2. (Abramsky). *If $t \to u$, then $\|t\| > \|u\|$. Thus every term is strongly normalizing, with the length of all reduction sequences bounded by $\|t\|$.*

Proof. Let t have width n. By structural induction, one shows that if $t \to u$, then $P_t(x) > P_u(x)$, for all $x \geq n$. If c is a variable or a basic constant, then the result is vacuously true since there are no redexes. If t is of the form $!s$, then any redex must occur in s. Let s^* be the term obtained from s by reducing this redex. By the induction hypothesis, we have $P_{!s}(x) = xP_s(x) > xP_{s^*}(x) = P_{!s^*}(x)$ for all $x \geq n$. If t is of the form $s_1 s_2$, then either the redex occurs in s_1, or the redex occurs in s_2, or $s_1 s_2$ is itself a redex. For the former two cases, suppose $s_1 \to s_1^*$. Then $P_{s_1 s_2}(x) = P_{s_1}(x) + P_{s_2}(x) > P_{s_1^*}(x) + P_{s_2}(x) = P_{s_1^* s_2}(x)$, by the induction hypothesis. Finally, in the latter case we check the two nontrivial cases:

1. $\mathbf{F}!s_1!s_2 \to !(s_1 s_2)$. Before reduction, we have $1 + xP_{s_1}(x) + xP_{s_2}(x)$, and after reduction, we have $x(P_{s_1}(x) + P_{s_2}(x))$.

2. $\mathbf{W}^n s_1!s_2 \to s_1 \underbrace{s_2 \cdots s_2}_{n}$. Before reduction, we have $1 + P_{s_1}(x) + xP_{s_2}(x)$, and after reduction, we have $P_{s_1}(x) + nP_{s_2}(x)$.

Therefore, $\|t\| = P_t(n) > P_u(n) \geq P_u(w(u)) = \|u\|$. □

Proposition 3.2.3. (Abramsky). *Reduction to normal form of PCL terms can be simulated by a Turing machine with polynomial overhead.*

To illustrate, consider the term $t = \mathbf{W}^2\mathbf{B}!xy$. In this case, $s(t) = 4$, $d(t) = 1$, $w(t) = 2$ and $P_t(x) = x + 3$, and so $\|t\| = 5$. In fact, this term normalizes in only two steps as follows: $\mathbf{W}^2\mathbf{B}!xy \to \mathbf{B}xxy \to x(xy)$.

Polytime Soundness of PCL

It is possible in PCL to give a system of representations as terms of various datatypes, e.g. unary and binary numerals. For example, unary numerals in PCL are defined as follows. For each natural number n, let $\mathbf{Z}_n$ denote a **BCK**-term representing the affine linear lambda term:

$$\lambda x_1 \ldots \lambda x_n \lambda y.x_1(\cdots(x_n y)\cdots)$$

(There is a standard procedure for translating back and forth between **BCK**-terms and affine linear lambda terms; see [22], for example.) We then define $\bar{n} \equiv \mathbf{W}^n\mathbf{Z}_n$. Then:

$$\bar{n}!xy \to \mathbf{Z}_n \underbrace{x \cdots x}_{n \text{ times}} y \to^* x(\cdots(xy)\cdots)$$

The first few terms together with their principal types schemes (see section on type inference) are as follows:

$$\begin{aligned}
\bar{0} &\equiv \mathbf{CK} : \beta \multimap \alpha \multimap \alpha \\
\bar{1} &\equiv \mathbf{W}^1\mathbf{I} : !\alpha \multimap \alpha \\
\bar{2} &\equiv \mathbf{W}^2\mathbf{B} : !(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha
\end{aligned}$$

Let $\mathbf{D}_1 \equiv \mathbf{B}(\mathbf{BB})^2$ and $\mathbf{D}_{n+1} \equiv \mathbf{BD}_n$. One can show by induction that $\mathbf{D}_n$ satisfies the equation $\mathbf{D}_n x_1 x_2 \cdots x_{n+2} x_{n+3} x_{n+4} = x_1 x_2 \cdots x_{n+2}(x_{n+3} x_{n+4})$. Then set $\mathbf{Z}_3 \equiv \mathbf{D}_1\mathbf{B}$ and $\mathbf{Z}_n \equiv \mathbf{D}_{n-2}\mathbf{Z}_{n-1}$ for $n > 3$. Again by induction one can show that $\mathbf{Z}_n$ satisfies the equation $\mathbf{Z}_n x_1 \cdots x_n y = x_1(\cdots(x_n y)\cdots)$. Therefore we define:

$$\begin{aligned}
\bar{3} &\equiv \mathbf{W}^3\mathbf{Z}_3 \\
\bar{4} &\equiv \mathbf{W}^4\mathbf{Z}_4 \\
\bar{5} &\equiv \mathbf{W}^5\mathbf{Z}_5 \\
&\vdots
\end{aligned}$$

Observe that for each natural number n, $d(\bar{n}) = 0$, $w(\bar{n}) = n$, and $s(\bar{n}) \in O(|n|)$. A partial recursive function $f : \mathbb{N} \to \mathbb{N}$ is said to be *numeralwise representable* in PCL

[2]This is a dickcissel in Smullyan's wonderful book [41].

if there is a term t satisfying:

$$\forall n \in \mathbb{N}. t\bar{n} = \bar{m} \iff f(n) = m.$$

The following result is due to Abramsky [2].

Proposition 3.2.4. (Abramsky). *Suppose we have a system of representations (say of binary numerals) as terms, which is of bounded depth, i.e. for some $d_0 \geq 1$, for all n, $d(\bar{n}) \leq d_0$, and also $s(\bar{n}), w(\bar{n}) \in O(|n|)$. Then any function numeralwise representable in PCL with respect to this system is polynomial time.*

Proof. Suppose t n.w.-represents f, then for all n,

$$\begin{aligned} \|t\bar{n}\| &= P_t(w) + P_{\bar{n}}(w) \\ &\leq w^{d(t)}s(t) + w^{d(\bar{n})}s(\bar{n}) \\ &\leq w^d(s(t) + s(\bar{n})) \\ &= w^d s \end{aligned}$$

where $d = \max(d(t), d_0)$, and s, w are $O(|n|)$. □

We shall therefore require that representations of datatypes in PCL have bounded depth in the above sense. Indeed, the representation of unary numerals above is of bounded depth. Representations of conditionals and lists in PCL can also be defined [2].

Next, we shall investigate the expressiveness of PCL.

Polytime Completeness of PCL

According to Abramsky [2], the key step in showing that PCL can represent all PTIME functions is to show how *polynomial-time iterations* can be represented. Given a polynomial $P(X)$ of degree k with coefficients in $\mathbb{N}$, and a term u, a PCL term $\mathsf{iter}_{P,u}$ can be defined such that, for all $n \in \mathbb{N}$,

$$\mathsf{iter}_{P,u} \underbrace{\bar{n} \cdots \bar{n}}_{k} x = u^{P(n)} x$$

where $u^0 x = x$ and $u^{i+1} x = u(u^i x)$ [2].

Suppose we are given a polynomial time function represented by a Turing machine which runs in polynomial time P. Of course, such an algorithm also runs on a polynomial space-bounded Turing machine (since at each step of the computation can explore at most one new cell). W.l.o.g. we assume $P(|n|) \geq |n|$ for all binary numerals n. Abramsky describes the high-level structure of the representation as follows [2]:

- We can program the transition function of a space-bounded Turing machine by purely affine means (i.e. using only **BCK** terms) using conditionals and a representation of lists.
- Given a binary numeral for n, we can convert it into an initial configuration for the Turing machine. Then a $P(|n|)$-length iteration is used to pad the tape out with blanks, making it large enough for the entire computation.
- We then then use another $P(|n|)$-length iteration with the Turing machine transition function to perform the computation.
- Finally, the result can be extracted from the final tape, again by purely affine means.

Therefore PCL characterizes PTIME:

Theorem 3.2.5. (Abramsky). *PCL is correct and complete for PTIME.*

We investigate next the problem of typing PCL terms, and its relationship to soft linear logic.

Type Inference

Logically, PCL corresponds to the $(\multimap, !)$-fragment of soft affine logic. Indeed, the principal type schemes of the basic PCL combinators are as follows. Note that we

associate types to the right.

$$\begin{aligned}
\mathbf{B} &: (\beta \multimap \gamma) \multimap (\alpha \multimap \beta) \multimap \alpha \multimap \gamma \\
\mathbf{C} &: (\alpha \multimap \beta \multimap \gamma) \multimap \beta \multimap \alpha \multimap \gamma \\
\mathbf{K} &: \alpha \multimap \beta \multimap \alpha \\
\mathbf{F} &: !(\alpha \multimap \beta) \multimap !\alpha \multimap !\beta \\
\mathbf{W}^n &: (\underbrace{\alpha \multimap \cdots \multimap \alpha}_{n} \multimap \beta) \multimap !\alpha \multimap \beta
\end{aligned}$$

The above types may be thought of as axiom schemes for a logic with only two rules, called Modus Ponens and Necessitation:

$$\frac{t : \alpha \multimap \beta \qquad u : \alpha}{tu : \beta} \qquad\qquad \frac{t : \alpha}{!t : !\alpha}$$

For example, one can derive $\mathbf{W}^2\mathbf{B} : !(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha$ as follows. Let $A \equiv \alpha \multimap \alpha$, then

$$\frac{\mathbf{W}^2 : (A \multimap A \multimap A) \multimap !A \multimap A \qquad \mathbf{B} : A \multimap A \multimap A}{\mathbf{W}^2\mathbf{B} : !A \multimap A}$$

where we have made the substitution $\alpha \mapsto A, \beta \mapsto A$ in the axiom scheme for $\mathbf{W}^2$ and the substitution $\alpha \mapsto \alpha, \beta \mapsto \alpha, \gamma \mapsto \alpha$ in the axiom scheme for the combinator $\mathbf{B}$. Similarly, one can derive $\mathbf{CKK} : \alpha \multimap \alpha$. Then $\mathbf{F!CKK} : !\alpha \multimap !\alpha$, as the following derivation shows.

$$\frac{\mathbf{F} : !(\alpha \multimap \alpha) \multimap !\alpha \multimap !\alpha \qquad \dfrac{\mathbf{CKK} : \alpha \multimap \alpha}{!\mathbf{CKK} : !(\alpha \multimap \alpha)}}{\mathbf{F!CKK} : !\alpha \multimap !\alpha}$$

where we have made the substitution $\alpha \mapsto \alpha, \beta \mapsto \alpha$ in the axiom scheme for $\mathbf{F}$.

It is a fact that all $\mathbf{BCK}$-combinators are typable [22], but this is not true for all $\mathbf{BCKW}$-combinators. Similarly, some PCL terms have types, while others do not. For example, one can check that the principal type schemes for the terms $\mathbf{I} \equiv \mathbf{CKK}$ and $\mathbf{W}^2\mathbf{B}$ are $\alpha \multimap \alpha$ and $!(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha$, respectively. However, the combinator $\mathbf{W}^2\mathbf{BI}$ is not typable; Indeed, if $\mathbf{W}^2\mathbf{BI} : \eta$, then $\mathbf{I} : \delta$ and $\mathbf{W}^2\mathbf{B} : \delta \multimap \eta$ and we need:

$$\begin{aligned}
\delta &\equiv \alpha \multimap \alpha \\
\delta \multimap \eta &\equiv !(\beta \multimap \beta) \multimap \beta \multimap \beta
\end{aligned}$$

It follows that $\alpha \multimap \alpha \equiv !(\beta \multimap \beta)$ which is impossible.

Conjecture 3.2.6. *The set of all typable PCL terms is recursively decidable.*

In fact, we have made some preliminary progress towards a type inference algorithm for PCL which will be reported elsewhere. This algorithm, for example, could be used to check whether Abramsky's encoding of PTIME algorithms is typable. We hope to return to these questions another time.

Subject Reduction

In this section we show that type-schemes are preserved by reduction. Its proof will need the following lemma.

Lemma 3.2.7. (Replacement). *Let $\mathcal{D}$ be a deduction of $X : \alpha$ in PCL. Let $\mathcal{D}_1$ be a subtree of $\mathcal{D}$ giving a deduction of $V : \gamma$, where V is a subterm of X. Let $\mathcal{D}_2$ be a deduction of $W : \gamma$ and let X^* be the term obtained by replacing V by W in X. Then* $\vdash_{\mathrm{PCL}} X^* : \alpha$.

Proof. See Lemma 14.23 in [23]. □

Theorem 3.2.8. (Subject Reduction). *If $X : \alpha$ and $X \to^* X'$ in PCL, then $X' : \alpha$.*

Proof. By the replacement lemma it is enough to assume that X is a redex and X' is its contractum. The cases $X \equiv \mathbf{B}UVW$, $X \equiv \mathbf{C}UVW$ and $X \equiv \mathbf{K}X'Y$ are standard and can be found in the literature (see [23], for example). The nonstandard cases are as follows.

Case 1: $X \equiv \mathbf{W}^n U!V$ and $X' \equiv U\underbrace{V \cdots V}_{n}$. A deduction $\mathcal{D}$ of $X : \alpha$ in PCL must have the following form:

$$\dfrac{\dfrac{\mathbf{W}^n : (\underbrace{\beta \multimap \cdots \multimap \beta}_{n} \multimap \alpha) \multimap !\beta \multimap \alpha \qquad U : \underbrace{\beta \multimap \cdots \multimap \beta}_{n} \multimap \alpha}{\mathbf{W}^n U : !\beta \multimap \alpha} \quad \begin{array}{c}\mathcal{D}_1 \\ \mathcal{D}_2 \\ V : \beta \\ \hline !V : !\beta\end{array}}{\mathbf{W}^n U!V : \alpha}$$

From this we can construct a deduction of $X' : \alpha$ using $\mathcal{D}_1$ and $\mathcal{D}_2$ (n times). For example, if $n = 2$, we can construct a deduction of $X' : \alpha$ as follows:

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1 \\ U : \beta \multimap \beta \multimap \alpha\end{matrix} \qquad \begin{matrix}\mathcal{D}_2 \\ V : \beta\end{matrix}}{UV : \beta \multimap \alpha} \qquad \begin{matrix}\mathcal{D}_2 \\ V : \beta\end{matrix}}{UVV : \alpha}$$

Case 2: $X \equiv \mathbf{F}!U!V$ and $X' \equiv !(UV)$. A deduction $\mathcal{D}$ of $X : !\alpha$ in PCL must have the following form:

$$\dfrac{\dfrac{\mathbf{F} : !(\beta \multimap \alpha) \multimap !\beta \multimap !\alpha \qquad \dfrac{\begin{matrix}\mathcal{D}_1 \\ U : \beta \multimap \alpha\end{matrix}}{!U : !(\beta \multimap \alpha)}}{\mathbf{F}!U : !\beta \multimap !\alpha} \qquad \dfrac{\begin{matrix}\mathcal{D}_2 \\ V : \beta\end{matrix}}{!V : !\beta}}{\mathbf{F}!U!V : !\alpha}$$

From this we can construct a deduction of $X' : !\alpha$ as follows:

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1 \\ U : \beta \multimap \alpha\end{matrix} \qquad \begin{matrix}\mathcal{D}_2 \\ V : \beta\end{matrix}}{UV : \alpha}}{!(UV) : !\alpha}$$

This completes the proof. □

Chapter 4

Multiplexor Categories

In this chapter, we give a categorical interpretation of (quantifier-free) intuitionistic multiplicative soft linear logic (IMSLL) and intuitionistic multiplicative soft affine logic (IMSAL). We call these categories *multiplexor categories* and *affine multiplexor categories*, respectively. As usual, we shall assume that an interpretation of the additive connective can be obtained by assuming that our categories have finite products. Given an (affine) multiplexor category B, our semantics takes the form of a structure-preserving functor from the syntactic category of IMSLL (resp. IMSAL) into B. Hence formulas are interpreted as objects in B and proofs are interpreted as arrows in B. Moreover, we show that for any proof π, if π^* denotes the corresponding cut-free proof, then $[\![\pi]\!] = [\![\pi^*]\!]$. In other words, multiplexor and affine multiplexor categories provide denotational semantics for IMSLL and IMSAL, respectively.

Next, we show that the soft exponential operator in each case can be interpreted canonically as a certain type of limit. We give a product formula for the soft exponential operator in each case which can be used if the category has sufficient limits[1]. It turns out that the well-known models of linear logic do have sufficient limits for this formula to be applied, and this often leads to a non-degenerate interpretation of the soft exponential, i.e. one that is not isomorphic to Girard's exponential. This is illustrated in the category of Girard's coherence spaces, the standard model of

[1]Abramsky gives the same formulas in unpublished work [2]. We were unaware of this work until the referees of our paper directed us to it [39].

linear logic. Finally, we explain why this interpretation does not suffice for a full completeness result.

4.1 Multiplexor Categories

Here are the definitions:

Definition 4.1.1. A *multiplexor category* consists of a triple $(\mathcal{C}, !, \{\mu_n\}_{n\geq 0})$, where (i) $\mathcal{C} = (\mathcal{C}, \otimes, \multimap, \mathbf{1})$ is an autonomous category (=symmetric monoidal closed category) with structure maps α, λ, ρ and τ (twist), (ii) $! = (!, b_2, b_0)$ is a symmetric monoidal endofunctor on $\mathcal{C}$ with $\mathcal{C}$-maps $b_2 : !A \otimes !B \to !(A \otimes B)$ and $b_0 : \mathbf{1} \to !\mathbf{1}$ (natural in A and B), and (iii) for each natural number n,

$$\mu_n : ! \stackrel{\cdot}{\to} (-)^n$$

is a monoidal natural transformation from ! to the n-fold tensor product functor, called a multiplexor of rank n. Implicit here is the commutativity of a number of diagrams. In particular, the definitions of symmetric monoidal functor and monoidal natural transformation can be found in Chapter 2. E.g. in addition to naturality of multiplexing, the following diagrams must commute:

$$\begin{array}{ccc} !A \otimes !B & \xrightarrow{b_2} & !(A \otimes B) \\ {\scriptstyle \mu_n \otimes \mu_n}\downarrow & & \downarrow{\scriptstyle \mu_n} \\ A^n \otimes B^n & \xrightarrow{\cong} & (A \otimes B)^n \end{array} \qquad \begin{array}{ccc} \mathbf{1} & \xrightarrow{b_0} & !\mathbf{1} \\ \| & & \downarrow{\scriptstyle \mu_n} \\ \mathbf{1} & \xrightarrow{\cong} & \mathbf{1}^n \end{array}$$

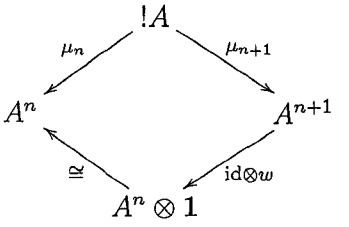
where $\cong$ means the canonical isomorphism (see Chapter 2). Moreover, for each natural number n and permutation σ in $S(n)$, the following diagram commutes:

$$\begin{array}{ccc} !A & \xrightarrow{\mu_n} & A^n \\ & {\scriptstyle \mu_n}\searrow & \downarrow{\scriptstyle \hat{\sigma}} \\ & & A^n \end{array} \tag{6}$$

where $\hat{\sigma} : A^n \to A^n$ is the corresponding canonical arrow. □

Not surprisingly, in the affine case, we need some additional diagrams to commute. Here is the definition.

Definition 4.1.2. An *affine multiplexor category* is a multiplexor category in which the tensor unit is also a terminal object in the category. Moreover, for each natural number n, we require that the following diagram commutes:

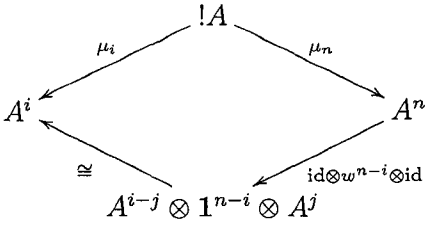


(7)

where $w_A : A \to 1$ is the unique map to the terminal object $\mathbf{1}$. □

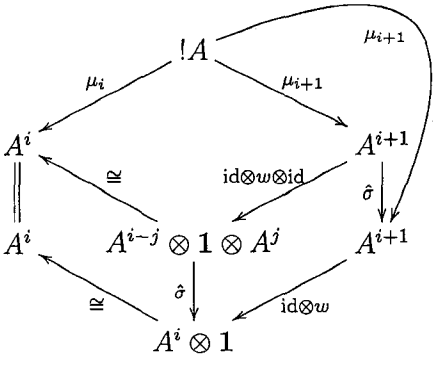
We have the following lemmas.

Lemma 4.1.3. *Let $\mathcal{C}$ be an affine multiplexor category. Then for all natural numbers $0 \leq j \leq i < n$, the following diagram commutes in $\mathcal{C}$:*



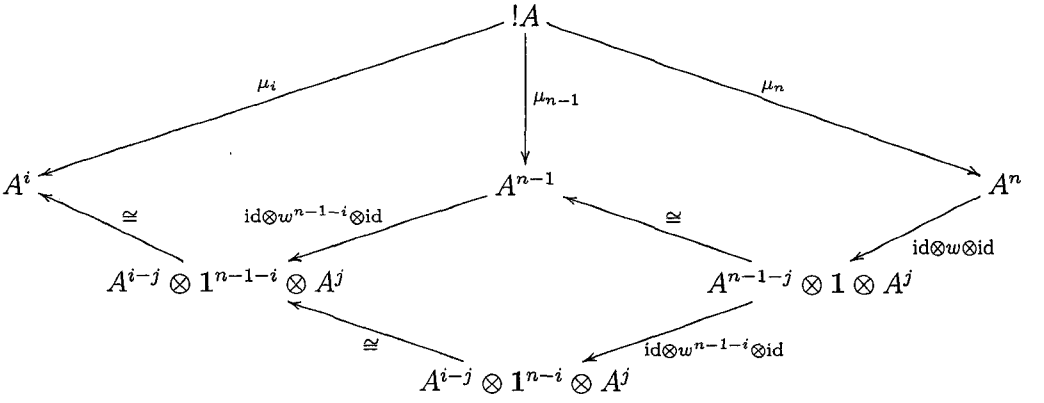
(8)

Proof. We prove by induction on $n - i \geq 1$. If $n = i + 1$, then consider the following diagram:



The outer path and right triangle commute by diagrams 6 and 7. And the two squares commute by naturality of $\hat{\sigma}$, and by the coherence theorem for symmetric monoidal

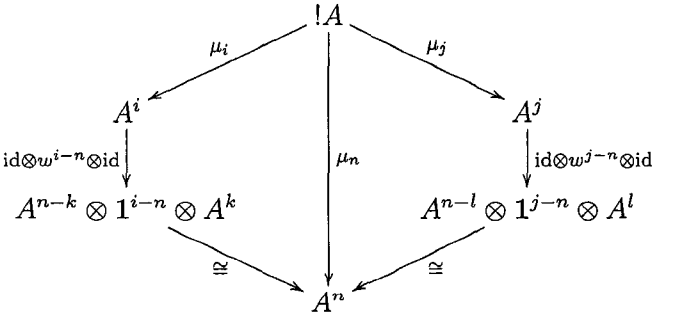
categories. It follows that the middle diamond commutes, proving the base case. Now assume $n - i > 1$ and consider the following diagram:



The bottom diamond commutes by naturality of the canonical isomorphism. The upper right diamond is just the base case, and the upper left diamond commutes by the induction hypothesis. Therefore the outer diamond commutes, which is what we needed to show. □

Lemma 4.1.4. *Let $\mathcal{C}$ be an affine multiplexor category. For any natural number n and any two parallel arrows $f, g : !A \to A^n$ using only multiplexing and weakening, we have $f = g = \mu_n$.*

Proof. By assumption, f and g must factor as in the outer paths in the following diagram:



By the previous lemma, both sides commute, hence $f = g = \mu_n$. □

Theorem 4.1.5. *An (affine) multiplexor category provides a denotational semantics for IMSLL (resp. IMSAL), in the sense that the interpretation of a proof is preserved under the respective cut-elimination procedure.*

Proof. We assume we have an environment ρ which assigns to each atom α an object of the category $\mathcal{C}$. Then any formula can be built up using the corresponding categorical constructors $\multimap, \otimes$ and ! on objects. A sequent $\Gamma = A_1, \ldots, A_n$ is interpreted as $A_1 \otimes \cdots \otimes A_n$ if $\Gamma \neq \emptyset$, and as the tensor unit $\mathbf{1}$ otherwise. Proofs are similarly built up using the corresponding categorical constructors on morphisms and canonical combinators as follows:

Logical Rule	**Uses in $\mathcal{C}$**
axiom	identity morphism
exchange	τ
cut	composition
$\multimap$R	curry(f)
$\multimap$L	ev
$\otimes$R	tensor on morphisms
$\otimes$L	-
$\mathbf{1}R$	id_1
$\mathbf{1}L$	ρ
!R	functoriality of !, b_0, b_2
!L	μ_n

Finally, it must be checked that the interpretation of a proof before each reduction (cut-elimination) step is the same as the interpretation after (see appendix A for the cut-elimination algorithm). The two non-standard cases are (i) !R versus !R, and (ii) !R versus !L. The first case is omitted, but uses only the fact that ! is a monoidal functor. For the second case, it suffices to assume $\Delta = \emptyset$. Let $\Gamma = A_1, \ldots, A_k$, $f : A_1 \otimes \cdots \otimes A_k \to A$ (if $k = 0$, then Γ is empty), and $g : A^n \to C$. Then the

cut-elimination step translates to the following diagram in $\mathcal{C}$:

$$\begin{array}{ccccccc}
!A_1 \otimes \cdots \otimes !A_k & \xrightarrow{b_*} & !(A_1 \otimes \cdots \otimes A_k) & \xrightarrow{!f} & !A \\
\downarrow{\scriptstyle \mu_n^k} & & \downarrow{\scriptstyle \mu_n} & & \downarrow{\scriptstyle \mu_n} \\
A_1^n \otimes \cdots \otimes A_k^n & \xrightarrow{\cong} & (A_1 \otimes \cdots \otimes A_k)^n & \xrightarrow{f^n} & A^n \xrightarrow{g} C
\end{array}$$

where the top path corresponds to the interpretation of the proof before reduction, and the bottom path corresponds to its interpretation after reduction. Both inside squares commute for any $k \geq 0$, since multiplexing is a monoidal natural transformation, therefore the entire diagram commutes and the interpretation is preserved. □

4.2 Multiplexor Functors and Transformations

Definition 4.2.1. A *multiplexor functor* $(F, b_2, b_0, c) : M \to M'$ consists of a symmetric monoidal functor $(F, b_2, b_0) : M \to M'$ and for each object a in M a morphism:

$$c(a) : !F(a) \to F(!a)$$

in M' which is natural in a. Moreover, for each natural number n, the following diagram must commute:

$$\begin{array}{ccc}
!F(a) & \xrightarrow{c} & F(!a) \\
\downarrow{\scriptstyle \mu_n} & & \downarrow{\scriptstyle F(\mu_n)} \\
F(a)^n & \xrightarrow{b^*} & F(a^n)
\end{array}$$

where b^* is the canonical iterated application of b_2 and b_0.

The composite of two multiplexor functors is a multiplexor functor.

Definition 4.2.2. A *multiplexor natural transformation* $\theta : (F, b_2, b_0, c) \to (G, b_2', b_0', d)$ is a monoidal natural transformation $\theta : (F, b_2, b_0) \to (G, b_2', b_0')$ such that the following diagram:

$$\begin{array}{ccc}
!F(a) & \xrightarrow{c} & F(!a) \\
\downarrow{\scriptstyle !\theta_a} & & \downarrow{\scriptstyle \theta_{!a}} \\
!G(a) & \xrightarrow{d} & G(!a)
\end{array}$$

in M'.

The composite of two multiplexor natural transformations is a multiplexor natural transformation.

4.3 Product Formulas

Our first examples of multiplexor categories come from models of linear logic. Multiplexing is a derived rule in the multiplicative exponential fragment of intuitionistic linear logic (IMELL) as the following derivation shows:

$$\begin{array}{c} A^{(n)} \vdash A^n \\ \vdots \\ !A^{(n)} \vdash A^n \\ \vdots \\ !A \vdash A^n \end{array}$$

In the above derivation, the first vertical dots represent n dereliction rule(s) (if $n \geq 1$), and the second vertical dots represent either weakening (if $n = 0$) or $n-1$ contraction rule(s) (if $n > 0$). Thus:

Lemma 4.3.1. *A linear category (see [8]) is an example of a multiplexor category.*

Proof. (sketch.) Diagram 6 (which is the outside path in the following diagram) in Def. 4.1.1 commutes because the following diagram commutes in any linear category:

$$\begin{array}{ccccc} !A & \xrightarrow{d_A^*} & (!A)^n & \xrightarrow{\epsilon_{!A}^n} & A^n \\ & \searrow^{d_A^*} & \downarrow{\scriptstyle \hat{\sigma}} & & \downarrow{\scriptstyle \hat{\sigma}} \\ & & (!A)^n & \xrightarrow{\epsilon_{!A}^n} & A^n \end{array}$$

for any permutation $\sigma \in S(n)$. The left triangle commutes by the commutative comonoid structure of $!A$ in a linear category. The right square commutes by naturality of $\hat{\sigma}$. □

Canonical interpretations of the soft exponential operator in IMSLL and IMSAL, which already explain the absence of contraction and digging, is given by the product formulas in the following two theorems (again, see footnote 1).

Theorem 4.3.2. *Let $\mathcal{C}$ be an autonomous category with countable products and equalizers of permutations (see below). Define:*

$$!A = \prod_n \mathcal{S}^n(A) = \mathbf{1} \times A \times (A \otimes_s A) \times (A \otimes_s A \otimes_s A) \times \cdots \quad . \tag{9}$$

where $A \otimes_s A$ is the following equalizer:

$$A \otimes_s A \xrightarrow{e_2} A \otimes A \underset{\tau}{\overset{id}{\rightrightarrows}} A \otimes A$$

and where τ is the twist map. The n-th symmetric power $\mathcal{S}^n(A)$ is defined analogously (as the equalizer of the set of permutations of n elements). Then $(\mathcal{C}, !)$ is a multiplexor category.

Theorem 4.3.3. *Let $\mathcal{C}$ be an autonomous category in which the tensor unit is terminal and the following limit exists. Define $!A$ to be the (projective) limit of the following diagram:*

$$\mathbf{1} \leftarrow A \leftarrow A \otimes_s A \leftarrow A \otimes_s A \otimes_s A \leftarrow \cdots \tag{10}$$

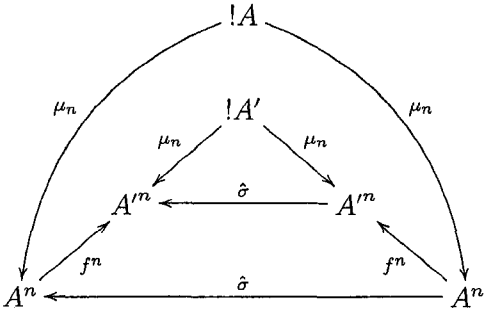
where for each natural number n, the map $\mathcal{S}^n(A) \leftarrow \mathcal{S}^{n+1}(A)$ is the unique map using the fact that $\mathbf{1}$ is terminal. Then $(\mathcal{C}, !)$ is an affine multiplexor category.

Proof. We shall prove in detail the first theorem; the proof of the second is similar. We define multiplexing μ_n to be the composite $e_n \circ p_n$, where p_n is the canonical projection onto $\mathcal{S}^n(A)$. Then observe that the following diagram commutes in $\mathcal{C}$:

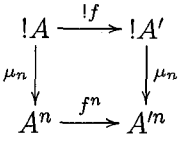
$$\begin{array}{ccccc} !A & \xrightarrow{p_n} & A^n & \xrightarrow{e_n} & A^n \\ & \searrow^{p_n} & \downarrow \mathrm{id} & & \downarrow \hat{\sigma} \\ & & A^n & \xrightarrow{e_n} & A^n \end{array}$$

for any natural number n and any permutation σ in $S(n)$. Hence diagram 6 in Def. 4.1.1 commutes.

Functoriality of !: Let $f : A \to A'$ be any morphism in $\mathcal{C}$. Then for any natural number n and permutation σ in $S(n)$, the following diagram commutes in $\mathcal{C}$:

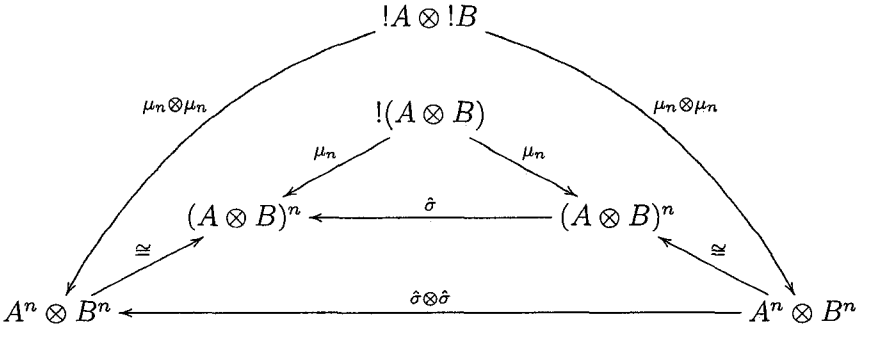


The bottom square commutes by naturality of $\hat{\sigma}$ and the outer and inner triangles commute by diagram 6. Since a similar diagram commutes for each natural number n and permutation $\sigma \in S(n)$, there is a unique map $!f : !A \to !A'$, making the following square commute in $\mathcal{C}$:



This argument constructs the arrow $!f$ and shows that multiplexing is a natural transformation. We leave it as an easy exercise to show that $!(g \circ f) = !g \circ !f$ and $!\mathrm{id_A} = \mathrm{id}_{!A}$, wherever the composite $g \circ f$ is defined.

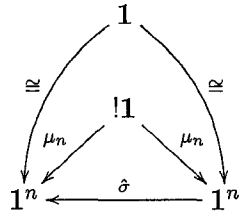
The map b_2: For each natural number n and permutation σ in $S(n)$ the following diagram commutes:



The bottom square commutes by the coherence theorem for symmetric monoidal categories, and the outer and inner triangles by diagram 6. Therefore, there is a unique map $b_2 : !A \otimes !B \to !(A \otimes B)$ making the following square commute:

$$\begin{array}{ccc} !A \otimes !B & \xrightarrow{b_2} & !(A \otimes B) \\ {\scriptstyle \mu_n \otimes \mu_n}\downarrow & & \downarrow{\scriptstyle \mu_n} \\ A^n \otimes B^n & \xrightarrow{\cong} & (A \otimes B)^n \end{array}$$

The map b_0: For each natural number n and permutation σ in $S(n)$ the following diagram commutes:

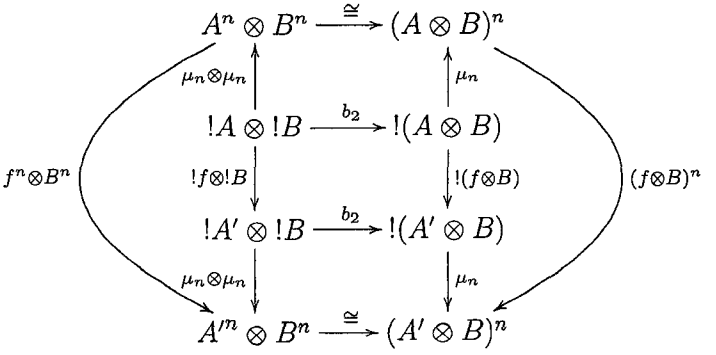


The outer triangle commutes by the coherence theorem for symmetric monoidal categories, and the inner triangle commutes diagram 6. Therefore, there is a unique map $b_0 : \mathbf{1} \to !\mathbf{1}$ making the following diagram commute:

$$\begin{array}{ccc} \mathbf{1} & \xrightarrow{b_0} & !\mathbf{1} \\ \| & & \downarrow{\scriptstyle \mu_n} \\ \mathbf{1} & \xrightarrow{\cong} & \mathbf{1}^n \end{array}$$

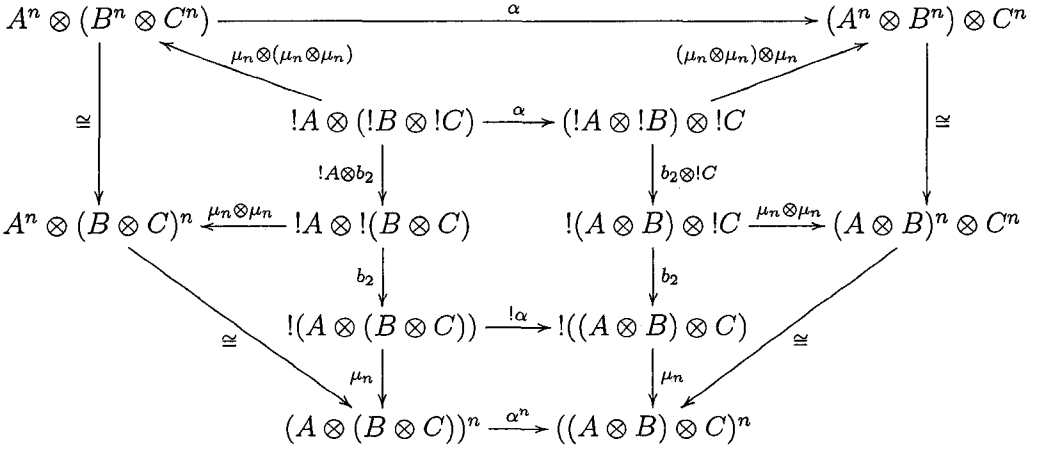
Therefore multiplexing is a monoidal natural transformation.

Naturality of b_2: Let $f : A \to A'$ in $\mathcal{C}$ and consider the following diagram:



Our goal is to show that the middle square commutes. Indeed, the surrounding four squares commute because μ_n is a monoidal natural transformation (which we have already shown). And the outer square commutes by naturality of the isomorphism. Since this argument works for any natural number n, it follows by Prop. 2.3.2 (see Chapter 2) that the middle square commutes. This proves naturality of b_2 in A. A similar argument shows that b_2 is natural in B.

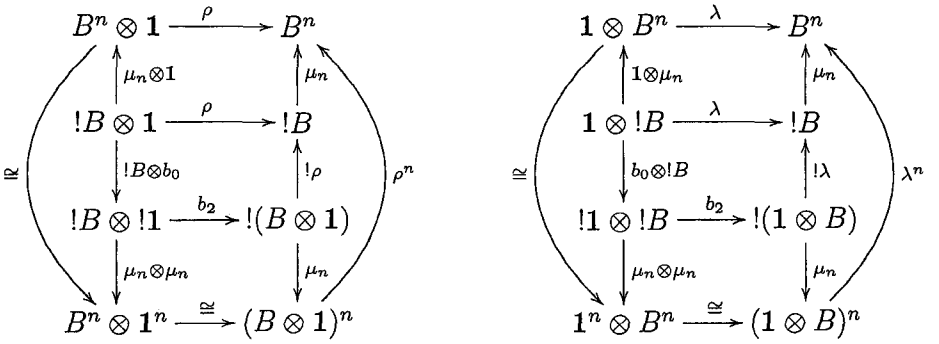
Coherence with α: For diagram 2 in the definition of symmetric monoidal functor (see Chapter 2) we must show that the middle square in the following diagram commutes:



The surrounding six squares commute by naturality of α (top square) and the fact that μ_n is a monoidal natural transformation. And the outer path commutes by the

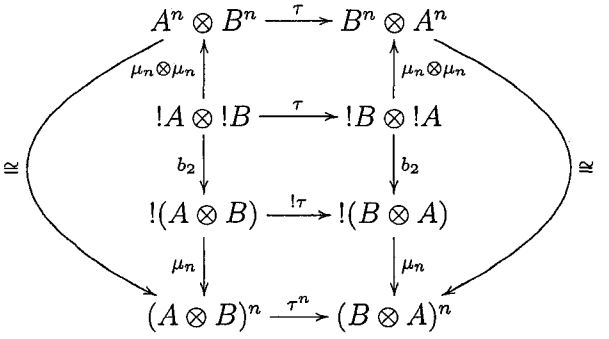
coherence theorem for symmetric monoidal categories. Since this argument works for any natural number n, the middle square commutes by Prop. 2.3.2 (see Chapter 2).

Coherence with ρ and λ: For diagrams 3 in the definition of symmetric monoidal functor we consider the following diagrams:



The bottom and side squares in each diagram commute because μ_n is a monoidal natural transformation. The top squares commute by naturality of ρ and λ and the outer squares commute by the coherence theorem for symmetric monoidal categories. Since this argument works for any natural number n, the middle square in each diagram commutes, again using Prop. 2.3.2 (see Chapter 2).

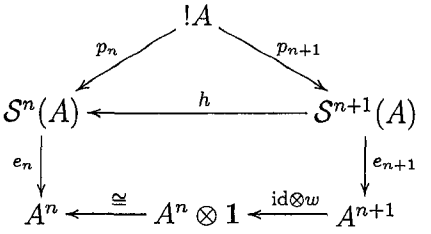
Coherence with τ: For diagram 4 in the definition of symmetric monoidal functor we consider the following diagram:



The bottom and side squares commute because μ_n is a monoidal natural transformation. The top square commutes by naturality of τ and the outer path commutes by

the coherence theorem for symmetric monoidal categories. Since this argument works for any natural number n, the middle square commutes, again using Prop. 2.3.2 (see Chapter 2).

This completes the proof for the first theorem. For the second, we need to consider, in addition, the following diagram:



The arrow h is the unique map making the bottom square commute. Moreover, the top triangle commutes by the structure of the limit diagram. Hence the outer diagram commutes, as desired. □

4.4 Coherence Spaces

Many of the standard models of IMELL have sufficient limits for the product formula to be applied. In many cases, this leads to a model in which the soft exponential has a separate interpretation from the usual (IMELL) one. As a concrete example, we shall use the product formula in the category of coherence spaces **Coh**, the standard model of linear logic. Later, we shall see another example in an AJM-style game semantics of IMELL.

Coherence Spaces

A *coherence space* is a pair $X = (|X|, \frown_X)$, where $|X|$ is a set called the web of X, and $\frown_X \subseteq |X| \times |X|$ is a binary, reflexive, symmetric relation. We write $x \frown_X y$ or $x \frown y \bmod X$ if $(x, y) \in \frown_X$, and in this case we say that x and y are coherent. Moreover, we write $x \frown_X y$ if $x \frown_X y$ and $x \neq y$ (*strict coherence*). The complement of the relation $\frown_X$ is *strict incoherence* $\smile_X$ and its reflexive closure is the *incoherence*

relation $\asymp_X$. A *clique* of X is a subset c of $|X|$ whose elements are pairwise coherent. A *multiclique* is a multiset whose underlying set is a clique.

Let X and Y be coherence spaces. We have the following constructions on coherence spaces:

- *negation:* $X^\perp = (|X|, \asymp_X)$
- *tensor:*

$$|X \otimes Y| = |X| \times |Y|$$
$$(x, y) \frown_{X \otimes Y} (x', y') \quad \text{iff} \quad x \frown_X x' \text{ and } y \frown_Y y'$$

- *with:*

$$|X \,\&\, Y| = |X| + |Y| = \{1\} \times |X| \cup \{2\} \times |Y|$$
$$(1, x) \frown_{X \& Y} (1, x') \quad \text{iff} \quad x \frown_X x'$$
$$(2, y) \frown_{X \& Y} (2, y') \quad \text{iff} \quad y \frown_Y y'$$
$$(1, x) \frown_{X \& Y} (2, y) \quad \text{for all } x \in |X| \text{ and } y \in |Y|$$

Then we set: $X \parr Y = (X^\perp \otimes Y^\perp)^\perp$, $X \oplus Y = (X^\perp \,\&\, Y^\perp)^\perp$ and $X \multimap Y = X^\perp \parr Y$. It follows that $|X \multimap Y| = |X| \times |Y|$ and:

$$(x, y) \frown_{X \multimap Y} (x', y') \quad \text{iff} \quad x \frown_X x' \text{ implies } y \frown_Y y'$$

The category **Coh** is now described as follows:

- Objects: Coherence spaces
- Morphisms: A morphism from X to Y is a clique of $X \multimap Y$.
- Composition: Usual relational composition.

With the constructions we have defined so far, the category **Coh** is a model of the multiplicative additive fragment of classical linear logic, or in categorical terms **Coh** is a $*$-autonomous category with finite products.

Exponential Structure

If X is a coherence space, then the interpretation of the linear logic exponential $!_{LL}$ is defined as $!_{LL}X = (\mathcal{M}_f(|X|), \subset_{!_{LL}X})$ where $\mathcal{M}_f(|X|)$ is the set of finite multicliques on $|X|$ and $u \subset_{!_{LL}X} v$ iff $u + v$ (multiset union) is a multiclique. On the other hand, using the product formula, we get a slightly different interpretation for the soft exponential operator $!_S$:

$$|!_S X| = \mathcal{M}_f(|X|)$$
$$\text{if } |u| = |v| \text{ then } u \subset_{!_S X} v \text{ iff } u + v \text{ is a multiclique}$$
$$u \subset_{!_S X} v \text{ for all } u, v \text{ such that } |u| \neq |v|$$

Consider the following relations:

1. *functoriality:* If f is a clique in $X \multimap Y$, then $!f : !X \multimap !Y$ is defined by:

$$!f = \{([x_1, \ldots, x_n], [y_1, \ldots, y_n]] \mid \forall i, (x_i, y_i) \in f \text{ and } [x_1, \ldots, x_n] \in |!X|\}$$

2. *monoidality:*

   (a) $\{(([x_1, \ldots, x_n], [y_1, \ldots, y_n]), [(x_1, y_1), \ldots, (x_n, y_n)]) \mid n \in \mathbb{N}, [x_1, \ldots, x_n] \in |!X|, [y_1, \ldots, y_n] \in |!Y|\} \sqsubset |!X| \times |!Y| \times |!(X \otimes Y)|$

   (b) $\{(\star, [n\star]) \mid n \in \mathbb{N}\} \sqsubset |\mathbf{1}| \times |!\mathbf{1}|$

3. *contraction:* $\{(u + v, (u, v)) | u + v \in |!X|\} \sqsubset |!X| \times (|!X| \times |!X|)$

4. *digging:* $\{(u_1 + \cdots + u_n, [u_1, \ldots, u_n]) \mid n \in \mathbb{N}, u_1 + \cdots + u_n \in |!X|\} \sqsubset |!X| \times |!!X|$

5. *multiplexing (of rank n):* $\{([x_1, \ldots, x_n], (x_1, \cdots, x_n)) \mid [x_1, \ldots, x_n] \in |!X|\} \sqsubset |!X| \times |X^n|$. In particular if $n = 0$, we have $\{([], \star)\} \sqsubset |!X| \times |\mathbf{1}|$

All of the above relations are cliques if $! = !_{LL}$, however contraction and digging are not if $! = !_S$. To see that contraction is not a clique when $! = !_S$, let $u, u', v, v' \in |!X|$ be such that $u \smile_X u'$, $|u| = |u'|$, $v = []$, and $v' \neq v$. Then $u + v \frown_{!_S X} u' + v'$ since $|u + v| = |u| < |u| + |v'| = |u' + v'|$, but $(u, v) \smile_{!_S X \otimes !_S X} (u', v')$. Therefore contraction is not valid in $(\mathbf{Coh}, !_S)$, and we have a model specific to soft linear logic. The argument for digging is similar.

Failure of Completeness

Next consider the following relation:

$$\{([x_1, y_1, \ldots, x_n, y_n], [(x_1, y_1), \ldots, (x_n, y_n)]) \mid n \in \mathbb{N}, [x_1, y_1, \ldots, x_n, y_n] \in |!X|\} \sqsubset |!X| \times |!(X \otimes X)|$$

The above relation is a clique under both interpretations, however the formula $!\alpha \multimap !(\alpha \otimes \alpha)$ is not provable in soft linear logic, for any atom α. In fact, it is easy to see that there will always be a canonical morphism $!A \to !(A \otimes A)$ in any category in which we apply the product formula interpretation of $!$. Therefore, this interpretation, though natural, is not good enough to obtain a completeness theorem. In the next chapter, we shall introduce a different method for constructing models of soft linear logic. This will lead to more refined models, and ultimately to a full completeness theorem.

Chapter 5

The S-Construction

Motivated by notions in complexity theory we introduce a finitary version of intuitionistic soft linear logic ISLL_f together with its corresponding categorical interpretation called a *finitary multiplexor category*. ISLL_f is obtained by replacing the soft exponential operator ! by an $\mathbb{N}$-indexed family of operators $!_0, !_1, !_2, \ldots$, together with corresponding bounded exponential logical rules. We'll see that a multiplexor category with exponential ! is trivially a finitary one by interpreting $!_n$ as !, for each natural number n. Our concern in this chapter will be with the reverse direction. We give a categorical method for constructing a multiplexor category $\mathcal{C}^s$ from a finitary one $\mathcal{C}$, called the *S-construction*. The S-construction (and its variants) will be used repeatedly in this thesis to construct more refined models of soft linear logic. These models lead to a stratified view of soft linear logic, in which a proof in IMSLL of rank n is thought of as an infinite family of proofs in ISLL_f starting at level n.

Given a proof π in ISLL, we can calculate its rank n (i.e. the maximal rank of multiplexors in π) and associate to it a corresponding indexed proof π_n in ISLL_f. Moreover, a semantics $F : \mathrm{ISLL}_f \to \mathcal{C}$ can be lifted to a semantics $F^s : \mathrm{ISLL} \to \mathcal{C}^s$ making the following diagram commute:

$$\begin{array}{ccc} \pi \in \mathrm{ISLL} & \longmapsto & \pi_n \in \mathrm{ISLL}_f \\ {\scriptstyle F^s}\downarrow & & \downarrow{\scriptstyle F} \\ \pi^* \in \mathcal{C}^s & \overset{|_n}{\longmapsto} & \pi_n^* \in \mathcal{C} \end{array}$$

where $|_n$ denotes the restriction of an arrow in $\mathcal{C}^s$ to its lowest level. We may even ask that for all $f \in \mathcal{C}^s$, if $f|_n = F(\pi_n) \in \mathcal{C}$ for some ISLL$_f$ proof π_n, then $f = \hat{F}(\pi)$, where π is the corresponding proof in ISLL obtained by forgetting the indices in π_n. This last condition is a type of *relative completeness*[1] result, since the fullness of the functor F implies the fullness of the functor F^s. These ideas will be elaborated on in this chapter.

5.1 Finitary Soft Linear Logic

Before introducing the construction, we first introduce a finitary version of soft linear logic (ISLL$_f$) with bounded multiplexing by replacing the soft exponential operator ! with an $\mathbb{N}$-indexed family of operators $!_0, !_1, !_2, \ldots$, together with the following exponential rules:

$$\frac{\Gamma \vdash A}{!_n\Gamma \vdash !_n A} \qquad \frac{\Gamma, A^{(n)} \vdash C}{\Gamma, !_{n+w}A \vdash C}$$

where w, n range over natural numbers. Intuitively, $!_n A$ is "cashable" for no more than n copies of datum A. This should remind the reader of the *bounded reuse operators* $!_x$ of BLL with exponential rules:

$$\frac{\Gamma \vdash C}{\Gamma, !_w A \vdash C} \qquad \frac{\Gamma, A \vdash C}{\Gamma, !_{1+w}A \vdash C} \qquad \frac{\Gamma, !_x A, !_y A \vdash C}{\Gamma, !_{x+y+w}A \vdash C} \qquad \frac{!_{\vec{y}}\Gamma \vdash A}{!_{x\vec{y}+w}\Gamma \vdash !_x A}$$

where x, y, w range over natural numbers [20]. Observe that in the latter system, the formula $!_{n+m}A \multimap !_n A \otimes !_m A$ is derivable; however in ISLL$_f$ it is not. In fact, we shall see models of ISLL$_f$ which invalidate this formula.

Lemma 5.1.1. ISLL$_f$ *satisfies cut elimination.*

The proof of this lemma is similar to the one for ISLL and is omitted. Observe that there is an obvious translation into ISLL which forgets the indices on the exponentials.

[1] At the time of writing, I am not sure of the relationship to Laurent and Tortora de Falco's notion of relative completeness in [30]. This has to be investigated further.

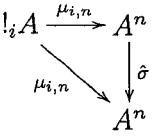
Finitary Multiplexor Categories

The categorical semantics developed for ISLL generalizes to the finitary case as follows.

Definition 5.1.2. A *finitary multiplexor category* consists of a triple of the form $(\mathcal{C}, \{!_i\}_{i\geq 0}, \{\mu_{i,n}\}_{i\geq n\geq 0})$, where $\mathcal{C} = (\mathcal{C}, \otimes, \multimap, \mathbf{1})$ is a symmetric monoidal closed category, $!_i = (!_i, b_{2,i}, b_{0,i})$ is a symmetric monoidal endofunctor on $\mathcal{C}$ for each natural number i, and for each pair of natural numbers $i \geq n \geq 0$,

$$\mu_{i,n} : !_i \stackrel{\cdot}{\to} (-)^n$$

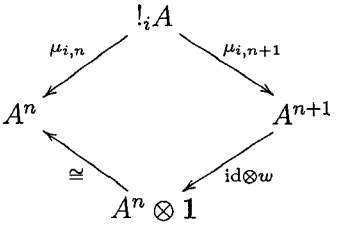
is a monoidal natural transformation from $!_i$ to the n-fold tensor product functor, called a multiplexor of rank (i, n). Moreover, we ask that for each pair of natural numbers $i \geq n \geq 0$, the following diagram commutes:



(11)

for each permutation σ in S_n.

Definition 5.1.3. An *affine finitary multiplexor category* is a finitary multiplexor category in which (i) the tensor unit $\mathbf{1}$ is terminal, and (ii) for all $i \geq n+1$, the following diagram commutes:



where $w_A : A \to \mathbf{1}$ is the unique map to the terminal object $\mathbf{1}$.

Example: $\mathbf{Rel}_f$

Any symmetric monoidal closed category with sufficient limits has a canonical structure of a finitary multiplexor category (and an affine one if the tensor unit is terminal) using the analogous product formulas introduced in the previous chapter. Another example is given as follows.

Let $\mathbf{Rel}_f$ be the full subcategory of $\mathbf{Rel}$ of all finite sets and relations with monoidal structure given by cartesian product $\times$ with unit $\{\star\}$, any one point set. For each natural number n, we define $!_nA$ to be the set of all finite multisets of elements A with cardinality at most n. For any natural number n, we have the following relations which are needed for modeling the exponentials in ISLL_f:

- *functoriality:* if $x \subseteq A \times B$, then $!_n x = \{[a_1, \ldots, a_k], [b_1, \ldots, b_k] |\ k \leq n \ \wedge\ \forall 1 \leq i \leq k.(a_i, b_i) \in x\} \subseteq !_nA \times !_nB$
- *multi-functoriality a):* $\{(([a_1, \ldots, a_k], [b_1, \ldots, b_k]), [(a_1, b_1), \ldots, (a_k, b_k)]) |\ k \leq n\} \subseteq (!_nA \times !_nB) \times !_n(A \times B)$
- *multi-functoriality b):* $\{(\star, [k\star]) |\ k \leq n\} \subseteq \mathbf{1} \times !_n\mathbf{1}$
- *multiplexing of rank (n, k):* $\{([a_1, \ldots, a_k], (a_1, \ldots, a_k))\} \subseteq !_nA \times A^k$. In particular, if $k = 0$, $\{([], \star)\} \subseteq !_nA \times \mathbf{1}$.

One can now check that $\mathbf{Rel}_f$ with these relations form a finitary multiplexor category. For example, let us check that

$$\begin{array}{ccc} !_2A & \xrightarrow{\mu_{2,2}} & A \otimes A \\ & \searrow^{\mu_{2,2}} & \downarrow{\scriptstyle \tau} \\ & & A \otimes A \end{array}$$

commutes in $\mathbf{Rel}_f$. In this case, we have $\mu_{2,2} = \{([a_1, a_2], (a_1, a_2)) |\ a_1, a_2 \in A\} \subseteq !_2A \times A^2$ and $\tau = \{((a_1, a_2), (a_2, a_1)) |\ a_1, a_2 \in A\} \subseteq A^2 \times A^2$. Then it is easy to see that $\tau \circ \mu_{2,2} = \{([a_2, a_1], (a_1, a_2)) |\ a_1, a_2 \in A\} \subseteq !_2A \times A^2$. Therefore $\mu_{2,2} = \tau \circ \mu_{2,2}$ since multisets are not ordered. The remaining details are omitted, but are similarly straightforward to check.

5.2 The S-Construction

Note that a multiplexor category is trivially a finitary multiplexor category in which we set $!_n := !$ for each natural number n. In this section, we are interested in the reverse direction. We give a categorical construction for extending a finitary multiplexor category $\mathcal{C}$ to a fully-fledged multiplexor category $\mathcal{C}^s$[2]. In this context, the category $\mathcal{C}$ will be called the base category.

The category $\mathcal{C}^s$ is described as follows:

- Objects: Objects are sequences of $\mathcal{C}$-objects indexed by $\mathbb{N}$. Notation: $A = (A_i) = (A_0, A_1, A_2, \dots)$.
- Morphisms: A morphism from (A_i) to (B_i) is an equivalence class of sequences $(k : f_i) = (k : f_k, f_{k+1}, \dots)$, where k is a natural number and for each $i \geq k$, $f_i : A_i \to B_i$ in $\mathcal{C}$. We say two sequences $(k : f_i)$ and $(k' : f'_i)$ are equivalent if there exists a natural number $m \geq k, k'$ such that for all $i \geq m$, $f_i = f'_i$ in $\mathcal{C}$. We shall abuse the notation and denote an equivalence class by one of its representatives.
- Composition: Let $f = (m, f_i) : A \to B$ and $g = (k, g_i) : B \to C$ be morphisms. We define composition by $f \circ g = (\max\{k, m\} : f_i \circ g_i) : A \to C$. It is easy to verify that this definition is well-defined: if $(m' : f'_i) \sim (m : f_i)$ and $(k', g'_i) \sim (k : g_i)$, then there exist natural numbers n_1 and n_2 such that for all $i \geq \max(n_1, n_2)$, $f_i = f'_i$ and $g_i = g'_i$ and $f_i \circ g_i = f'_i \circ g'_i$. Therefore $f \circ g \sim f' \circ g'$, as claimed. The identity on (A_i) is $(0 : id_{A_i})$.

Associativity and the unit laws follow by the corresponding diagrams in the base category $\mathcal{C}$. Therefore $\mathcal{C}^s$ is a category. There is an obvious embedding $J : \mathcal{C} \to \mathcal{C}^s$ defined by:

$$\begin{aligned} A &\mapsto (A, A, A, \dots) \\ f &\mapsto (0 : f) \end{aligned}$$

which preserves all the structure, as we shall see. We will show:

[2]As far as I can tell this is not an adjunction.

Proposition 5.2.1. *$\mathcal{C}^s$ is a multiplexor category which inherits the monoidal structure of the base category $\mathcal{C}$. Moreover, if $\mathcal{C}$ is finitely complete or affine then so is $\mathcal{C}^s$.*

Tensor

Let $A = (A_i)$ and $B = (B_i)$ be two objects in $\mathcal{C}^s$. We define their tensor product $A \otimes B$ componentwise by $(A_i \otimes B_i)$ and the tensor unit by $J(\mathbf{1}) = (\mathbf{1}, \mathbf{1}, \dots)$, where $\mathbf{1}$ is the tensor unit in $\mathcal{C}$. Observe that if $\mathbf{1}$ is terminal in $\mathcal{C}$, then $J(\mathbf{1})$ is terminal in $\mathcal{C}^s$. If $f = (m : f_i) : A \to C$ and $g = (k : g_i) : B \to D$ are morphisms in $\mathcal{C}^s$, we define $f \otimes g : A \otimes B \to C \otimes D$ by $(\max(k, m), f_i \otimes g_i)$. Again, it is straightforward to check that this definition is well-defined. The structure maps are given by sequences of the corresponding structure maps in the base category $\mathcal{C}$. Finally, the coherence diagrams follow by the corresponding commuting diagrams in $\mathcal{C}$. Therefore, $\mathcal{C}^s$ is a symmetric monoidal category.

Linear Implication

Let $A = (A_i)$ and $B = (B_i)$ be two objects in $\mathcal{C}^s$. We define a linear implication object $A \multimap B$ componentwise by $(A_i \multimap B_i)$. The evaluation map ev is defined as $(0 : \mathrm{ev}_i) : ((A_i \multimap B_i) \otimes A_i) \to (B_i)$, where ev_i is the corresponding evaluation map in $\mathcal{C}$. Given any morphism $f = (k : f_i) : (X_i \otimes A_i) \to (B_i)$, we have a morphism defined by $\hat{f} = (k : \hat{f}_i) : (X_i) \to (A_i \multimap B_i)$, where for each $i \geq k$, $\hat{f}_i$ is the unique map in $\mathcal{C}$ such that $\mathrm{ev}_i \circ (\hat{f}_i \otimes id_{A_i}) = f_i$. (Note that the assignment $f \mapsto \hat{f}$ is well-defined.) Then

$$\mathrm{ev} \circ (\hat{f} \otimes id_A) = (k : \mathrm{ev}_i \circ (\hat{f}_i \otimes id_{A_i})) = (k : f_i) = f$$

Finally, if $g = (m : g_i)$ satisfies

$$\mathrm{ev} \circ (g \otimes id_A) = (m : \mathrm{ev}_i \circ (g_i \otimes id_{A_i})) = (k : f_i) = f$$

then there exists a natural number $n \geq k, m$ such that for all $i \geq n$, $\mathrm{ev}_i \circ (g_i \otimes id_{A_i}) = f_i$, and so $g_i = \hat{f}_i$ by the uniqueness of the $\hat{f}_i$ in the base category $\mathcal{C}$. Hence $g = \hat{f}$, establishing the uniqueness of $\hat{f}$. Therefore $\mathcal{C}^s$ is a symmetric monoidal closed category.

Binary Products

Suppose that the base category has binary products, and let $A = (A_i)$ and $B = (B_i)$ be a pair of objects in $\mathcal{C}^s$. For each level $i \geq 0$, the product $A_i \times B_i$ exists in $\mathcal{C}$ and comes equipped with projection maps $p_i^1 : A_i \times B_i \to A_i$ and $p_i^2 : A_i \times B_i \to B_i$. We therefore define the product in $\mathcal{C}^s$ by $A \times B = (A_i \times B_i)$ together with the projection morphisms $p^1 = (0 : p_i^1) : A \times B \to A$ and $p^2 = (0 : p_i^2) : A \times B \to B$. Let $C = (C_i)$ be an object, and $f = (k, f_i) : C \to A$ and $g = (n, g_i) : C \to B$ be morphisms in $\mathcal{C}^s$. Then for all $i \geq \max(k, n)$, there exists a unique $\mathcal{C}$-map $h_i : C_i \to A_i \times B_i$ such that $p_i^1 \circ h_i = f_i$ and $p_i^2 \circ h_i = g_i$. Then the morphism $h = (\max(k, n) : h_i) : C \to A \times B$ is such that $p^1 \circ h = f$ and $p^2 \circ h = g$ in $\mathcal{C}^s$. Finally, if $h' = (l, h_i') : C \to A \times B$ is such that $p^1 \circ h' = f$ and $p^2 \circ h' = g$ in $\mathcal{C}^s$, then there exists a natural number $m \geq \max(k, n, l)$ such that for all $i \geq m$, $p_i^1 \circ h_i' = f_i$ and $p_i^2 \circ h_i' = g_i$, and so $h_i' = h_i$ by uniqueness of h_i in $\mathcal{C}$. Therefore, $h' = h$ in $\mathcal{C}^s$ and $\mathcal{C}^s$ has binary products. More generally, one can show that $\mathcal{C}^s$ is finitely complete if $\mathcal{C}$ is. However, we conjecture that the analogous result for infinite limits is not in general true.

Soft Exponential

The soft exponential operator is defined as follows:

$$\begin{aligned} !(A_i) &= (!_0 A_0, !_1 A_1, !_2 A_2, \dots) \\ !(k : f_i) &= (k : !_i f_i) \end{aligned}$$

This definition is indeed functorial as $!id_A = !(0 : id_A) = (0 : !_i id_A) = (0 : id_{!_i A}) = id_{!A}$ and $!f \circ !g = !(k : f_i) \circ !(m : g_i) = (k : !_i f_i) \circ (m : !_i g_i) = (\max(k, m), !_i f_i \circ !_i g_i) = (\max(k, m), !_i(f_i \circ g_i)) = !(\max(k, m), f_i \circ g_i) = !(f \circ g)$. Moreover, we have the following maps: (i) The map $b_0 : (\mathbf{1}) \to (!_i \mathbf{1})$ is defined as $(0 : b_{0,i})$, where $b_{0,i} : \mathbf{1} \to !_i \mathbf{1}$ is the corresponding map in $\mathcal{C}$; (ii) The map $b_2 : (!_i A_i \otimes !_i B_i) \to (!_i (A_i \otimes B_i))$ is defined as $(0 : b_{2,i})$, where $b_{2,i} : !_i A_i \otimes !_i B_i \to !_i (A_i \otimes B_i)$ is the corresponding map in $\mathcal{C}$, and (iii) for each natural number n, multiplexing $\mu_n : (!_i A_i) \to (A_i)^n$ is defined as $(n : \mu_{i,n})$, where $\mu_{i,n} : !_i A_i \to A_i^n$ is the corresponding map in $\mathcal{C}$. Multiplexing is a natural

transformation because for any $f = (k : f_i) : (A_i) \to (B_i)$, we have

$$\begin{aligned}
\mu_n \circ !f &= (n : \mu_{i,n}) \circ (k : !_i f_i) \\
&= (\max(n,k) : \mu_{i,n} \circ !_i f_i) \\
&= (\max(n,k) : f_i^n \circ \mu_{i,n}) \\
&= (k : f_i^n) \circ (n : \mu_{i,n}) \\
&= f^n \circ \mu_n
\end{aligned}$$

Finally, the commutativity of the coherence diagrams is immediate by the corresponding diagrams in the base category together with the equivalence relation on morphisms. Therefore $\mathcal{C}^s$ is a multiplexor category.

Uniformity

Recall in theory of circuit families a uniformity condition is added which intuitively means that circuits at different levels represent the same algorithm. So far, the levels in the S-construction are completely independent. We would like to fix this by placing restrictions on morphisms to achieve some level of dependence. We shall call any such restriction a *uniformity condition.* The uniformity condition(s) we put on morphisms will in general depend on the chosen base category $\mathcal{C}$. For example, recall the category $\mathbf{Rel}_f$ of finite sets and relations which we showed to be a finitary multiplexor category, and consider the subcategory of $\mathbf{Rel}_f^s$ consisting of nested relations:

$$(n : x_i) = x_n \subseteq x_{n+1} \subseteq x_{n+2} \subseteq \cdots$$

Observe that nested relations are preserved under composition and each of the constructors $\otimes, \multimap, \&, !$. Moreover, one can check that the interpretation of any ISLL proof is nested. Therefore we obtain a more refined model by restricting to the subcategory of $\mathbf{Rel}_f^s$ consisting of nested relations. We shall see other uniformity conditions in the models that we introduce later. In each case, we seek a (natural) semantic condition which ensures that morphisms at different levels are closely related. Ideally, the uniformity condition(s) should be strong enough to give relative completeness.

5.3 A Variant of the S-Construction

In this section, we shall investigate an important class of models in which the base category is either **Set**-like or **Rel**-like in a way to be made precise shortly. For this restricted class of finitary multiplexor categories, the S-construction can be simplified and this leads to a large class of models, including one which has recently appeared in the literature [30].

Set-like models

Recall the simple set-theoretic interpretation of linear logic which is defined on objects by $(A \multimap B)_* = A_* \Rightarrow B_*$, $(A \otimes B)_* = (A \,\&\, B)_* = A_* \times B_*$, $(!A)_* = A_*$ and $\mathbf{1}_* = \{\star\}$. Suppose we have a finitary multiplexor category $\mathcal{C}$ together with a forgetful (or faithful) functor $U : \mathcal{C} \to \mathbf{Set}$, which preserves the structure strictly. If A is an object in $\mathcal{C}$, we write $U(A) = |A|$. Then $|A \multimap B| = |B|^{|A|}$, $|A \otimes B| = |A \,\&\, B| = |A| \times |B|$, $|!_n A| = |A|$, and $|\mathbf{1}| = \{\star\}$, and similarly for morphisms. In this case, the S-construction can be simplified and the category $\mathcal{C}^s$ is described as follows:

- Objects: Objects are sequences of $\mathcal{C}$-objects indexed by $\mathbb{N}$ (notation: $(A_i) = (A_0, A_1, A_2, \ldots)$) with constant underlying set $|A|$
- Morphisms: A morphism from (A_i) to (B_i) is a function f from $|A|$ to $|B|$ such that there exists some $n \in \mathbb{N}$ with for any $i \geq n$, f is a morphism from A_i to B_i in $\mathcal{C}$
- Composition: Usual composition of functions.

Observe that we no longer need to explicitly track the index n in the morphisms, and an equivalence relation on sequences is not necessary either. The constructions we defined before (i.e. tensor, product, linear implication, soft exponential) carry over to this setting, and we again have that $\mathcal{C}^s$ is a multiplexor category. Moreover, as a uniformity condition on morphisms f, we may ask that for all natural numbers n: if f is a morphism at level n then this implies that f is a morphism at level $n+1$. For an example of this type of model, we refer the reader to the realizability model for ISAL_2 we construct in the next chapter.

Rel-like models

Recall the relational interpretation of linear logic which is defined on objects by $(A \multimap B)_* = (A \otimes B)_* = A_* \times B_*$, $(A \& B)_* = A_* + B_*$ (disjoint union), $(!A)_* = \mathcal{M}_f(A_*)$ (the set of finite multisets of elements of A_*) and $\mathbf{1}_* = \{\star\}$. Suppose we have a finitary multiplexor category $\mathcal{C}$ together with a forgetful (or faithful) functor $U : \mathcal{C} \to \mathbf{Rel}$, which preserves the structure strictly. If A is an object in $\mathcal{C}$, we write $U(A) = |A|$. Then $|A \multimap B| = |A \otimes B| = |A| \times |B|$, $|A \,\&\, B| = |A| + |B|$ $|!_n A| = \mathcal{M}_f(|A|)$, and $|\mathbf{1}| = \{\star\}$, and similarly for morphisms. In this case, the S-construction can be simplified and the category $\mathcal{C}^s$ is described as follows:

- Objects: Objects are sequences of $\mathcal{C}$-objects indexed by $\mathbb{N}$ (notation: $(A_i) = (A_0, A_1, A_2, \ldots)$) with constant underlying set $|A|$
- Morphisms: A morphism from (A_i) to (B_i) is a relation f on $|A| \times |B|$ such that there exists some $n \in \mathbb{N}$ with for any $i \geq n$, f is a morphism from A_i to B_i in $\mathcal{C}$
- Composition: Usual relational composition of sets.

Again, observe that we no longer need to explicitly track the index n in the morphisms, and an equivalence relation on sequences is not necessary either. The constructions we defined before (i.e. tensor, product, linear implication, soft exponential) carry over to this setting, and we again have that $\mathcal{C}^s$ is a multiplexor category. Moreover, as a uniformity condition on morphisms f, we may ask that for all natural numbers n: if f is a morphism at level n then this implies that f is a morphism at level $n+1$. We shall present a nice example of this type of model, independently discovered by Olivier Laurent and Lorenzo Tortora de Falco [30], in the next section.

Relative Completeness

There is another desirable property which is shared by both the **Set**-like and **Rel**-like models discussed above and it is this: for all $f \in \mathcal{C}^s$, if there exists a level n (i.e. natural number) such that $f = F(\pi_n)$ (i.e. f is the denotation of a proof π_n in

ISLL_f), then $f = F^s(\pi_n^*)$, where π_n^* is the corresponding proof in ISLL. Consequently, if ISLL_f is fully-complete with interpretation functor F, then ISLL is fully-complete with interpretation functor F^s. We shall call this property *relative completeness*. It will be important when we prove full-completeness for IMSAL.

5.4 Obsessional Cliques

As an example of a **Rel**-like model, we describe in this section a model recently proposed by Oliver Laurent and Lorenzo Tortora de Falco in [30]. The two relevant categories described in that paper are (i) **ORel**, a relational model of so-called *obsessional cliques*, and (ii) **SRel**, a stratified version of the latter. It turns out that this is an instance of the variant of the S-construction described above, i.e. **ORel** is a finitary multiplexor category and $\mathbf{SRel} = \mathbf{ORel}^s$, but this was not known to the authors at the time. Therefore our methods provide an alternative description of this model.

The Category ORel

Recall the category **Rel** of sets and relations with monoidal and exponential structure described in the previous section. Following [30], we shall call *cliques* of a set A the subsets of A, in the spirit of coherence spaces. The category **ORel** is described as follows:

- Objects: $\mathbb{N}$-sets. (A $\mathbb{N}$-set is given by a set A together with a monoid action $(k, a) \mapsto a^{(k)}$ from $\mathbb{N}^* \times A$ to A[3])
- Morphisms: A morphism from $(A, \_^{(\cdot)})$ to $(B, \_^{(\cdot)})$ is a relation (clique) $R \subseteq A \times B$ which *preserves the action*, i.e.

$$\forall k \in \mathbb{N}^*, \forall (a, b) \in R.(a^{(k)}, b^{(k)}) \in R$$

  Such a relation is called an *obsessional clique*.

[3] In other words, $a^{(1)} = a$ and $a^{(kk')} = (a^{(k)})^{(k')}$.

- Composition: usual relational composition.

The monoidal structure is inherited from **Rel** with pointwise action on $A \times B$ and the only possible action on $\mathbf{1} = \{\star\}$. If $t \in \mathbb{N}$, we define $!_t A$ as the $\mathbb{N}$-set with underlying set $\mathcal{M}_f(A)$ and the action on $!_t A$ is given by:

$$[a_1, \ldots, a_n]^{(k)} = \begin{cases} [a_1^{(k)}, \ldots, a_n^{(k)}] & \text{if } n \leq t \\ [ka_1^{(k)}, \ldots, ka_n^{(k)}] & \text{if } n > t \end{cases}$$

where ka is short for $a, \ldots, a$ (k times). It turns out that **ORel** is a finitary multiplexor category and the forgetful functor U : **ORel** $\to$ **Rel** preserve the structure strictly, hence we can apply the S-construction. An interesting observation is that bounded contraction (cf. the BLL-like exponential rules at the beginning of this chapter) can be refuted in this model. (Also observe that there are no maps $!_i A \multimap !_j A$ for $i > j$.)

The Category SRel

The category **SRel** is described as follows:

- Objects: an object is a set A with a family of actions indexed by $\mathbb{N}$ giving it $\mathbb{N}$-set structures $(A_n)_{n \in \mathbb{N}}$
- Morphisms: a morphism from $(A_n)_{n \in \mathbb{N}}$ to $(B_n)_{n \in \mathbb{N}}$ is a relation (clique) $R \subseteq A \times B$ such that there exists some $t \in \mathbb{N}$ with for any $n \geq t$, R is an obsessional clique of $A_n \times B_n$.
- Composition: usual relational composition.

There is an embedding J : **ORel** $\to$ **SRel** defined on objects by $|J(A)| = |A|$ and choosing all the actions of the family to be the action of A. In a similar way, constructions on objects in **ORel** can be turned into constructions on objects of **SRel** by applying them for each level n. As a specific construction, we define:

$$!(A_n)_{n \in \mathbb{N}} = (!_n A_n)_{n \in \mathbb{N}}$$

In conclusion, **SRel** is a multiplexor category, and can be viewed as the category obtained by applying the S-construction to the base category **ORel**. As remarked in [30], one could use the category **Coh** in place of **Rel** also.

5.5 The S-Construction Revisited

Let $\mathcal{C}$ be a category. Let $Ob(\mathcal{C})$ be the category defined as follows[4]:

- Objects: the objects of $\mathcal{C}$.
- Arrows: $\hom(a, b) = \{*\}$. (hence all objects are isomorphic)

Now define the following collection of functors:

$$\begin{aligned} F_1 &: \prod_{i\geq 0} \mathcal{C} \to Ob(\mathcal{C}) \times \prod_{i\geq 0} \mathcal{C} \\ F_2 &: Ob(\mathcal{C}) \times \prod_{i\geq 0} \mathcal{C} \to Ob(\mathcal{C}) \times Ob(\mathcal{C}) \times \prod_{i\geq 0} \mathcal{C} \\ F_3 &: \cdots \end{aligned}$$

defined by

$$\begin{aligned} F_i(a_0, a_1, a_2, \ldots) &= (a_0, a_1, a_2, \ldots) \\ F_i(\underbrace{*, \ldots, *}_{i-1}, f_{i-1}, f_i, f_{i+1}, \ldots) &= (\underbrace{*, \ldots, *}_{i}, f_i, f_{i+1}, \ldots) \end{aligned}$$

Now consider the following sequence of categories and functors:

$$\prod_{i\geq 0} \mathcal{C} \xrightarrow{F_1} Ob(\mathcal{C}) \times \prod_{i\geq 0} \mathcal{C} \xrightarrow{F_2} Ob(\mathcal{C}) \times Ob(\mathcal{C}) \times \prod_{i\geq 0} \mathcal{C} \xrightarrow{F_3} \cdots$$

Theorem 5.5.1. *$\mathcal{C}^s$ is a colimit of the above sequence.*

Proof. The injections $I_0, I_1, \ldots$ are defined by the identity on objects and on arrows:

$$I_i(\underbrace{*, \ldots, *}_{i}, f_i, f_{i+1}, \ldots) = [(i : f_i)]$$

[4]This is also known as the "chaotic" category of $\mathcal{C}$

The fact that, for each i, $I_{i+1} \circ F_{i+1} = I_i$ follows by the equivalence relation on sequences of arrows in the S-construction.

Now suppose we have another category $\mathcal{P}$ and injections $J_0, J_1, \ldots$ such that, for each i, $J_{i+1} \circ F_{i+1} = J_i$. Then we define a unique functor $H : \mathcal{C}^s \to \mathcal{P}$ as follows:

$$\begin{aligned} H(a_0, a_1, \ldots) &= J_0(a_0, a_1, \ldots) \\ H([(i : f_i)]) &= H(I_i(\underbrace{*, \ldots, *}_{i}, f_i, f_{i+1}, \ldots)) \\ &= J_i(\underbrace{*, \ldots, *}_{i}, f_i, f_{i+1}, \ldots)) \end{aligned}$$

This is well-defined because if $(j; f_j) \sim (i; f_i)$, where $k \geq i, j$ is the index after which the terms are equal, then:

$$\begin{aligned} H([(i : f_i)]) &= J_i(\underbrace{*, \ldots, *}_{i}, f_i, f_{i+1}, \ldots) \\ &= J_k F_k \cdots F_{i+1}(\underbrace{*, \ldots, *}_{i}, f_i, f_{i+1}, \ldots) \\ &= J_k(\underbrace{*, \ldots, *}_{k}, f_k, f_{k+1}, \ldots) \\ &= J_k F_k \cdots F_{j+1}(\underbrace{*, \ldots, *}_{j}, f_j, f_{j+1}, \ldots) \\ &= J_j(\underbrace{*, \ldots, *}_{j}, f_j, f_{j+1}, \ldots) \\ &= H([(j : f_j)]) \end{aligned}$$

Next we should check that H is a functor (w.l.o.g. suppose $j \geq i$):

$$\begin{aligned} H([(i : f_i)] \circ [(j : g_j)]) &= H([(j : f_j \circ g_j)]) \\ &= J_j(\underbrace{*, \ldots, *}_{j}, f_j \circ g_j, f_{j+1} \circ g_{j+1}, \ldots) \\ &= J_j(\underbrace{*, \ldots, *}_{j}, f_j, f_{j+1}, \ldots) \circ J_j(\underbrace{*, \ldots, *}_{j}, g_j, g_{j+1}, \ldots) \\ &= J_j F_j \cdots F_{i+1}(\underbrace{*, \ldots, *}_{i}, f_i, f_{i+1}, \ldots) \circ J_j(\underbrace{*, \ldots, *}_{j}, g_j, g_{j+1}, \ldots) \\ &= J_i(\underbrace{*, \ldots, *}_{i}, f_i, f_{i+1}, \ldots) \circ J_j(\underbrace{*, \ldots, *}_{j}, g_j, g_{j+1}, \ldots) \\ &= H([(i : f_i)]) \circ H([(j : g_j)]) \end{aligned}$$

$$\begin{aligned} H([(i : id_{a_i})]) &= J_i(\underbrace{*, \ldots, *}_{i}, id_{a_i}, id_{a_{i+1}}, \ldots) \\ &= id_{J_i(a_0, a_1, \ldots)} \\ &= id_{J_0(a_0, a_1, \ldots)} \\ &= id_{H(a_0, a_1, \ldots)} \end{aligned}$$

This completes the proof. □

In particular, we have:

$$\hom_{\mathcal{C}^s}((a_i), (b_i)) \cong \varinjlim_{n} \prod_{i \geq n} \hom_{\mathcal{C}}(a_i, b_i)$$

Chapter 6

A Realizability Model for ISAL_2

In this chapter, we give a realizability model of ISAL_2 using the S-construction of the previous chapter[1]. The resulting categorical model is similar to the one for BLL in [25]. The main result is a new proof that all polynomial time algorithms representable in ISLL_2 (with the original encoding by Lafont in [28]) are polytime. This result follows by the soundness of the interpretation. It would be interesting to compare this approach to recent work done in [15].

6.1 Preliminaries

An untyped lambda term is *affine linear* (or simply *linear*) if each variable, free or bound, appears at most once up to α-congruence. E.g. $\lambda x.x$, $\lambda x.y$, $(\lambda x.x)(\lambda x.y)$ are linear; the term $\lambda x.xx$ is not. Observe that any affine linear term t is strongly normalizable in less than $|t|$ steps where $|t|$ is the size of the term, because contracting a redex strictly reduces the size of a term. The runtime of the computation leading to the normal form is therefore $O(|t|^2)$. We will henceforth use the expression *affine lambda term* for an untyped affine lambda term which is in normal form. If s, t are affine linear lambda terms then their application st is defined as the normal form

[1]Rather than construct a base category of realizability sets with *binary* relations (on $\Lambda_a \times |A|$) first, and then perform the S-construction on top of that, we combine the two steps by considering realizability sets with *ternary* relations (on $\mathbb{N} \times \Lambda_a \times |A|$).

of the lambda term st. Observe that the application st can be computed in time $O((|s|+|t|)^2)$. We shall write Λ_a for the set of closed affine lambda terms.

6.2 A Realizability Category $\mathcal{B}^s$

The category $\mathcal{B}^s$ is described as follows:

- Objects: A *realizability set* is a pair $(|A|, \Vdash_A)$, where $|A|$ is a set and $\Vdash_A$ is a ternary relation $\Vdash_A \subseteq \mathbb{N} \times \Lambda_a \times |A|$.
- Morphisms: A *morphism* from $(|A|, \Vdash_A)$ to $(|B|, \Vdash_B)$ is a function $f : |A| \to |B|$ such that there exists a natural number N and a sequence of terms $\{e_i\}_{i \geq N}$ (in Λ_a) such that:

  $$i, t \Vdash_A a \qquad \text{implies} \qquad i, e_i t \Vdash_B f(a)$$

  for all $t \in \Lambda_a$ and $a \in |A|$ and $i \geq N$. In this case we say that $\{e_i\}_{i \geq N}$ *witnesses* or *realizes* the function f. Moreover, we require that there exists a polynomial $p(i)$ (over $\mathbb{N}$) such that $|e_i| \leq p(i)$ for all $i \geq N$. We call $p(i)$ a bounding polynomial.
- Composition: Usual function composition.

The identity morphism on $A = (|A|, \Vdash_A)$ is the identity function on $|A|$ witnessed by terms $\{\lambda u.u\}_{i \geq 0}$ and bounded by a constant. Given morphisms $f : (|A|, \Vdash_A) \to (|B|, \Vdash_B)$ and $g : (|B|, \Vdash_B) \to (|C|, \Vdash_C)$ witnessed by $\{e_i\}_{i \geq N_1}$ and $\{e'_i\}_{i \geq N_2}$, and bounded by polynomials $p(i)$ and $q(i)$, respectively, the composite function gf may be witnessed by $\{\lambda z.e'_i(e_i z)\}_{i \geq \max(N_1, N_2)}$ and bounded by the polynomial $p(i)+q(i)+c$, where c is a constant. The resulting category we denote as $\mathcal{B}^s$.

Tensor

Let $A = (|A|, \Vdash_A)$ and $B = (|B|, \Vdash_B)$ be realizability sets. Their tensor product is defined as $A \otimes B = (|A| \times |B|, \Vdash_{A \otimes B})$, where

$$i, t \Vdash_{A \otimes B} (a, b) \qquad \text{iff} \qquad t = T t_1 t_2 \quad \text{where} \quad i, t_1 \Vdash_A a \quad \text{and} \quad i, t_2 \Vdash_B b$$

where $T = \lambda xyf.fxy$. For morphisms f witnessed by $\{d_i\}_{i\geq N_1}$ and bounded by $p(i)$, and g witnessed by $\{e_i\}_{i\geq N_2}$ and bounded by $q(i)$, the function $f \times g$ can be witnessed by the sequence of terms $\{\lambda p.p(\lambda t_1 t_2.T(d_i t_1)(e_i t_2))\}_{i\geq \max(N_1,N_2)}$ and is bounded by the polynomial $p(i) + q(i) + c$, where c is a constant. The tensor unit is defined as $1 = (\{*\}, \Vdash_1)$ such that $i, \lambda x.x \Vdash_1 *$ for all natural numbers i, which is also a terminal object. Then:

- Associativity $(a,(b,c)) \mapsto ((a,b),c)$ is realized by $\{\lambda p.p(\lambda xv.v(\lambda yz.T(Txy)z))\}_{i\geq 0}$. The inverse $((a,b),c)) \mapsto (a,(b,c))$ is realized by $\{\lambda p.p(\lambda xv.x(\lambda yz.T(yT(zv))))\}_{i\geq 0}$.
- The structural map $(*,b) \mapsto b$ is realized by the terms $\{\lambda p.p(\lambda xy.y)\}_{i\geq 0}$. The inverse map is realized by $\{\lambda x.T(\lambda u.u)x\}_{i\geq 0}$. And similarly for the structural map $(a,*) \mapsto a$.
- Symmetry $(a,b) \mapsto (b,a)$ is realized by $\{\lambda p.p(\lambda xy.Tyx)\}_{i\geq 0}$. The same witness for the inverse.

All of the above are bounded by constant polynomials and so they are realizability morphisms. Note that the coherence diagrams are immediate by the corresponding diagrams of underlying functions and sets. Therefore $\mathcal{B}^s$ is a symmetric monoidal category.

Linear Implication

Let $A = (|A|, \Vdash_A)$ and $B = (|B|, \Vdash_B)$ be realizability sets. We define a linear function object $A \multimap B = (|A| \Rightarrow |B|, \Vdash_{A\multimap B})$ where

$$i, e \Vdash_{A\multimap B} f \quad \text{iff} \quad \text{whenever } \; i, t \Vdash_A a \; \text{ then } \; i, et \Vdash_B f(a)$$

The evaluation map $(f,a) \mapsto f(a)$ is realized by $\{\lambda z.z(\lambda xy.xy)\}_{i\geq 0}$ and is bounded by a constant. If $f : C \otimes A \to B$ is witnessed by $\{e_i\}_{i\geq N}$ and bounded by polynomial $p(i)$, then $\lambda(f) : C \to A \multimap B$ given by $\lambda(f)(c)(a) = f(c,a)$ is witnessed by $\{\lambda uv.e_i(\lambda k.kuv)\}_{i\geq N}$ and bounded by polynomial $p(i) + c$, where c is a constant. Therefore $\mathcal{B}^s$ is a symmetric monoidal closed category.

Binary Products

The category $\mathcal{B}^s$ has cartesian products given by $A \times B = (|A| \times |B|, \Vdash_{A\times B})$ where

$$i, e \Vdash_{A\times B} (a, b) \quad \text{iff} \quad i, e\mathbf{tt} \Vdash_A a \quad \text{and} \quad i, e\mathbf{ff} \Vdash_B b$$

Projections $A \times B \to A$ and $A \times B \to B$ are realized by $\{\lambda e.e\mathbf{tt}\}_{i\geq 0}$ and $\{\lambda e.e\mathbf{ff}\}_{i\geq 0}$, respectively, and bounded by constant polynomials. If $f : C \to A$ and $g : C \to B$ are realized by $\{d_i\}_{i\geq n_1}$ and $\{e_i\}_{i\geq n_2}$ and bounded by $p(i)$ and $q(i)$, respectively, then the function $\langle f, g\rangle : C \to A \times B$ defined by $\langle f, g\rangle(c) = (f(c), g(c))$ is realized by $\{\lambda xt.(Dtd_ie_i)x\}_{i\geq \max(n_1,n_2)}$, where D is the combinator $\lambda txy.txy$. This is bounded by a polynomial in $O(p(i) + q(i))$.

Soft Exponential

Let $A = (|A|, \Vdash_A)$ be a realizability set. We define the soft exponential as $!A = (|A|, \Vdash_{!A})$, where

$$i, t \Vdash_{!A} a \quad \text{iff} \quad t = \lambda f.ft_1t_2 \ldots t_i \quad \text{where} \quad \forall j \in \{1, \ldots, i\}.i, t_j \Vdash_A a$$

This extends to a functor. Let f be a morphism of realizability sets witnessed by $\{e_i\}_{i\geq N}$ and bounded by polynomial $p(i)$. We define $!f = f$, as functions. This can be witnessed by the following terms:

$$\{\lambda p.p(\lambda x_1x_2 \ldots x_if.f(e_ix_1)(e_ix_2) \cdots (e_ix_i))\}_{i\geq N}$$

which are bounded by the polynomial $ip(i) + 2i + c$, where c is a constant. (Note that this is our most "expensive" definition.)

Moreover, we have:

- The map $b_0 : \mathbf{1} \to !\mathbf{1}$ $(* \mapsto *)$ can be witnessed by the terms $\{\lambda xf.fI^{i-1}x\}_{i\geq 0}$.
- The map $b_2 : !A \otimes !B \to !(A \otimes B)$ $((a, b) \mapsto (a, b))$ can be witnessed by the terms:

$$\{\lambda p.p(\lambda uv.u(\lambda x_1 \ldots x_i.v(\lambda y_1 \ldots y_if.f(Tx_1y_1) \ldots (Tx_iy_i))))\}_{i\geq 0}$$

- For each $n \geq 2$, multiplexing $m_n(A) : !A \to A^n$ is defined by $m_n(A)(a) = (\underbrace{a, \ldots, a}_{n \text{ times}})$, and can be realized by the following terms:

$$\{\lambda p.p(\lambda x_1 \ldots x_k.T(\cdots(T(Tx_1x_2)x_3)\cdots)x_n)\}_{k \geq n}$$

- For $n = 1$, multiplexing $m_1(A) : !A \to A$ is defined by $m_1(A)(a) = a$ and can be witnessed by $\{\lambda p.p(\lambda x_1 \ldots x_k.x_1)\}_{k \geq 1}$.

- For $n = 0$, multiplexing $m_0(A) : !A \to \mathbf{1}$ is the unique map to the one point set $\{*\}$ which can be witnessed by the terms $\{\lambda p.p(\lambda x_1 \ldots x_k.I)\}_{k \geq 0}$.

Each of the above can be bounded by a linear polynomial and so is a realizability morphism. Moreover, all of the coherence diagrams hold by the corresponding diagrams of sets and functions. All together, we have:

Proposition 6.2.1. *$\mathcal{B}^s$ is an affine multiplexor category.*

6.3 Interpretation of $ISAL_2$

Following [25] we shall assume that our ambient set theory is constructive, so that we have a set of sets U containing the natural numbers $\mathbb{N}$, closed under product, function space, and U-indexed products. This allows one to interpret types as sets in the following way: given a formula A and an environment ρ which assigns sets to all *free* second-order variables occurring in A, we obtain a set-theoretic interpretation $[\![A]\!]^{Set}_{\rho}$ as follows. We assume without loss of generality that all bound variables in A

and in ρ are distinct from each other.

$$\begin{aligned}
[\![\alpha]\!]_\rho^{Set} &= \rho(\alpha) \\
[\![A \otimes B]\!]_\rho^{Set} &= [\![A]\!]_\rho^{Set} \times [\![B]\!]_\rho^{Set} \\
[\![A \& B]\!]_\rho^{Set} &= [\![A]\!]_\rho^{Set} \times [\![B]\!]_\rho^{Set} \\
[\![A \multimap B]\!]_\rho^{Set} &= [\![A]\!]_\rho^{Set} \Rightarrow [\![B]\!]_\rho^{Set} \\
[\![\mathbf{1}]\!]_\rho^{Set} &= \{*\} \\
[\![!A]\!]_\rho^{Set} &= [\![A]\!]_\rho^{Set} \\
[\![\forall \alpha.A]\!]_\rho^{Set} &= \prod_{C \in U} [\![A]\!]_{\rho[\alpha \mapsto C]}^{Set}
\end{aligned}$$

To every proof π of a sequent $A_1, \ldots, A_n \vdash B$ and environment ρ, we can assign a set theoretic function

$$[\![\pi]\!]_\rho^{Set} : [\![A_1 \otimes \cdots \otimes A_n]\!]_\rho^{Set} \to [\![B]\!]_\rho^{Set}$$

by induction on derivations, in the obvious way. For universal application, the following lemma is useful and is proved by structural induction.

Lemma 6.3.1. *Let A be a $ISAL_2$ formula possibly containing the (free) second-order variable α. Let B be another $ISAL_2$ formula. Then:*

$$[\![A[B/\alpha]]\!]_\rho^{Set} = [\![A]\!]_{\rho[\alpha \mapsto [\![B]\!]_\rho^{Set}]}^{Set}$$

Proof. A few of the cases are as follows:
Case 1. Variable $A \equiv \alpha$:

$$\begin{aligned}
[\![A[B/\alpha]]\!]_\rho^{Set} &= [\![\alpha[B/\alpha]]\!]_\rho^{Set} \\
&= [\![B]\!]_\rho^{Set} \\
&= \rho[\alpha \mapsto [\![B]\!]_\rho^{Set}](\alpha) \\
&= [\![\alpha]\!]_{\rho[\alpha \mapsto [\![B]\!]_\rho^{Set}]}^{Set}
\end{aligned}$$

Case 2. Variable $A \equiv \beta \neq \alpha$:

$$\begin{aligned}
[\![A[B/\alpha]]\!]_\rho^{Set} &= [\![\beta[B/\alpha]]\!]_\rho^{Set} \\
&= [\![\beta]\!]_\rho^{Set} \\
&= \rho(\beta) \\
&= \rho[\alpha \mapsto [\![B]\!]_\rho^{Set}](\beta) \\
&= [\![\beta]\!]_{\rho[\alpha \mapsto [\![B]\!]_\rho^{Set}]}^{Set}
\end{aligned}$$

Case 3. Universal quantification:

$$\begin{aligned}
[\![(\forall\beta.A)[B/\alpha]]\!]_\rho^{Set} &= [\![\forall\beta.A[B/\alpha]]\!]_\rho^{Set} \\
&= \prod_{C \in U} [\![A[B/\alpha]]\!]_{\rho[\beta \mapsto C]}^{Set} \\
&= \prod_{C \in U} [\![A]\!]_{\rho[\beta \mapsto C, \alpha \mapsto [\![B]\!]_{\rho[\beta \mapsto C]}^{Set}]}^{Set} \\
&= [\![\forall\beta.A]\!]_{\rho[\alpha \mapsto [\![B]\!]_\rho^{Set}]}^{Set}
\end{aligned}$$

□

Let A be a ISAL$_2$ formula and ρ be an environment which assigns a realizability set to each free second-order variable occurring in A. If C is a realizability set, we shall write $|\rho|$ for the mapping $\alpha \mapsto |C|$. By induction on A we define a realizability set $[\![A]\!]_\rho^{\mathcal{B}^s}$ such that

$$|[\![A]\!]_\rho^{\mathcal{B}^s}| = [\![A]\!]_{|\rho|}^{Set}$$

The defining clauses are as follows:

$$\begin{aligned}
[\![\alpha]\!]_\rho^{\mathcal{B}^s} &= \rho(\alpha) \\
[\![A \otimes B]\!]_\rho^{\mathcal{B}^s} &= [\![A]\!]_\rho^{\mathcal{B}^s} \otimes [\![B]\!]_\rho^{\mathcal{B}^s} \\
[\![A \& B]\!]_\rho^{\mathcal{B}^s} &= [\![A]\!]_\rho^{\mathcal{B}^s} \times [\![B]\!]_\rho^{\mathcal{B}^s} \\
[\![A \multimap B]\!]_\rho^{\mathcal{B}^s} &= [\![A]\!]_\rho^{\mathcal{B}^s} \multimap [\![B]\!]_\rho^{\mathcal{B}^s} \\
[\![\mathbf{1}]\!]_\rho^{\mathcal{B}^s} &= (\{*\}, \Vdash_{\mathbf{1}}) \\
[\![!A]\!]_\rho^{\mathcal{B}^s} &= ![\![A]\!]_\rho^{\mathcal{B}^s} \\
[\![\forall\alpha.A]\!]_\rho^{\mathcal{B}^s} &= (\prod_{C \in U} [\![A]\!]_{|\rho[\alpha \mapsto C]|}^{Set}, \Vdash_{[\![\forall\alpha.A]\!]_\rho^{\mathcal{B}^s}})
\end{aligned}$$

where

$$i,t \Vdash_{[\![\forall\alpha.A]\!]^{\mathcal{B}^s}_\rho} f \quad \text{iff} \quad \forall C \in U.\ i,t \Vdash_{[\![A]\!]^{\mathcal{B}^s}_{\rho[\alpha\mapsto C]}} f_C$$

Again, for universal application, we use the following lemma which can be proved by structural induction.

Lemma 6.3.2. *Let A be a ISAL$_2$ formula possibly containing the (free) second-order variable α. Let B be another ISAL$_2$ formula. Then:*

$$[\![A[B/\alpha]]\!]^{\mathcal{B}^s}_\rho = [\![A]\!]^{\mathcal{B}^s}_{\rho[\alpha\mapsto[\![B]\!]^{\mathcal{B}^s}_\rho]}$$

Proof. The proof is similar to the proof of the previous lemma. We give here only one non-trivial case.
Universal quantification:

$$\begin{aligned}
(i,t,f) \in \Vdash_{[\![(\forall\beta.A)[B/\alpha]]\!]^{\mathcal{B}^s}_\rho} \quad \text{iff} \quad & (i,t,f) \in \Vdash_{[\![\forall\beta.A[B/\alpha]]\!]^{\mathcal{B}^s}_\rho} \\
\text{iff} \quad & \forall C \in U.\ (i,t,f_C) \in \Vdash_{[\![A[B/\alpha]]\!]^{\mathcal{B}^s}_{\rho[\beta\mapsto C]}} \\
\text{iff} \quad & \forall C \in U.\ (i,t,f_C) \in \Vdash_{[\![A]\!]^{\mathcal{B}^s}_{\rho[\beta\mapsto C,\alpha\mapsto[\![B]\!]^{\mathcal{B}^s}_{\rho[\beta\mapsto C]}]}} \\
\text{iff} \quad & (i,t,f) \in \Vdash_{[\![\forall\beta.A]\!]^{\mathcal{B}^s}_{\rho[\alpha\mapsto[\![B]\!]^{\mathcal{B}^s}_\rho]}}
\end{aligned}$$

□

The following theorem is the main result of this section.

Theorem 6.3.3. (Soundness). *Let π be a proof of a sequent $A_1,\ldots,A_n \vdash B$ in ISAL$_2$. Let ρ be an environment binding all second-order free variables in the sequent. Then the set-theoretic function*

$$[\![\pi]\!]^{Set}_{|\rho|} : [\![A_1 \otimes \cdots \otimes A_n]\!]^{Set}_{|\rho|} \to [\![B]\!]^{Set}_{|\rho|}$$

is a morphism of realizability sets from $[\![A_1 \otimes \cdots \otimes A_n]\!]^{\mathcal{B}^s}_\rho$ to $[\![B]\!]^{\mathcal{B}^s}_\rho$.

Proof. By induction on ISAL$_2$ derivations. For universal abstraction: by induction, we have for each $C \in U$, a function:

$$f_C : [\![A_1 \otimes \cdots \otimes A_n]\!]^{Set}_{|\rho|[\alpha\mapsto C]} \to [\![B]\!]^{Set}_{|\rho|[\alpha\mapsto C]}$$

Since α is not free in Γ, for each $C \in U$, we have a function:

$$f_C : [\![A_1 \otimes \cdots \otimes A_n]\!]^{Set}_{|\rho|} \to [\![B]\!]^{Set}_{|\rho|[\alpha \mapsto C]}$$

and hence a function:

$$f : [\![A_1 \otimes \cdots \otimes A_n]\!]^{Set}_{|\rho|} \to \prod_{C \in U} [\![B]\!]^{Set}_{|\rho|[\alpha \mapsto C]}$$

This is a realizability morphism: By induction, there is a sequence of terms $\{e_i\}_{i \geq N}$ that witness f_C, for all $C \in U$. We claim that this sequence of realizers also witnesses f. Indeed, let $i \geq N$ and $i, t \Vdash_{[\![A_1 \otimes \cdots \otimes A_n]\!]^{\mathcal{B}^s}_\rho} a$. Then for all $C \in U$, $i, e_i t \Vdash_{[\![B]\!]^{\mathcal{B}^s}_{\rho[\alpha \mapsto C]}} f_C(a)$, hence $i, e_i t \Vdash_{[\![\forall \alpha. B]\!]^{\mathcal{B}^s}_\rho} f(a)$.

For universal application, we need a realizability morphism $[\![\forall \alpha. A]\!]^{\mathcal{B}^s}_\rho \to [\![A[B/\alpha]]\!]^{\mathcal{B}^s}_\rho$. By the above lemmas, this is the same as demanding a realizability there be a morphism $[\![\forall \alpha. A]\!]^{\mathcal{B}^s}_\rho \to [\![A]\!]^{\mathcal{B}^s}_{\rho[\alpha \mapsto [\![B]\!]^{\mathcal{B}^s}_\rho]}$. There is a function $\prod_{C \in U} [\![A]\!]^{Set}_{|\rho|[\alpha \mapsto C]} \to [\![A]\!]^{Set}_{|\rho|[\alpha \mapsto [\![B]\!]^{\mathcal{B}^s}_\rho]}$ by instantiating C with $[\![B]\!]^{\mathcal{B}^s}_\rho$. This can be witnessed by a sequence of identity terms. The other cases are standard and omitted. □

6.4 PTIME Correctness of $ISLL_2$

The following are the data types of (tally) natural numbers, strings (i.e. dyadic lists), and booleans in $ISLL_2$ [28]:

$$\begin{aligned} \mathbf{N} &= \forall \alpha.!(\alpha \multimap \alpha) \multimap (\alpha \multimap \alpha) \\ \mathbf{S} &= \forall \alpha.!((\alpha \multimap \alpha) \,\&\, (\alpha \multimap \alpha)) \multimap (\alpha \multimap \alpha) \\ \mathbf{B} &= \forall \alpha.(\alpha \,\&\, \alpha) \multimap \alpha \end{aligned}$$

Under the set-theoretic interpretation, we have e.g:

$$\begin{aligned} [\![\mathbf{S}]\!]^{Set}_\rho &= \prod_{C \in U} (C \Rightarrow C) \times (C \Rightarrow C) \Rightarrow (C \Rightarrow C) \\ [\![\mathbf{B}]\!]^{Set}_\rho &= \prod_{C \in U} C \times C \Rightarrow C \end{aligned} \tag{12}$$

Tally natural numbers can be encoded in affine lambda terms by $\ulcorner 0 \urcorner = \mathbf{tt} \otimes \mathbf{tt}$ and $\ulcorner n + 1 \urcorner = \mathbf{ff} \otimes \ulcorner n \urcorner$ where $\mathbf{tt} = \lambda xy.x$ and $\mathbf{ff} = \lambda xy.y$. Strings may be encoded as

$\ulcorner \epsilon \urcorner = (\lambda xyz.x) \otimes (\lambda xy.x)$ and $\ulcorner 0w \urcorner = (\lambda xyz.y) \otimes \ulcorner w \urcorner$ and $\ulcorner 1w \urcorner = (\lambda xyz.z) \otimes \ulcorner w \urcorner$. Observe that $\ulcorner n \urcorner$ and $\ulcorner w \urcorner$ can be computed in linear time from n, respectively w, and vice versa.

The following are some useful examples of realizability sets:

1. The realizability set of tally natural numbers is given by $\mathbf{N}_x = (\mathbb{N}, \Vdash_{\mathbf{N}_x})$ where

$$n, \ulcorner n \urcorner \Vdash_{\mathbf{N}_x} n \qquad \text{iff} \qquad n \in \mathbb{N}$$

2. Then given any polynomial $p(x)$, we can define a realizability set of natural numbers $\mathbf{N}_{p(x)} = (\mathbb{N}, \Vdash_{\mathbf{N}_{p(x)}})$ where

$$i, \ulcorner n \urcorner \Vdash_{\mathbf{N}_{p(x)}} n \qquad \text{iff} \qquad p(i), \ulcorner n \urcorner \Vdash_{\mathbf{N}_x} n$$

   Note that the successor function S on $\mathbb{N}$ is a realizability morphism from $\mathbf{N}_x$ to $\mathbf{N}_{x+1}$. Observe that a generic (i.e. realized at every level) morphism $f : \mathbf{N}_x \to \mathbf{N}_{p(x)}$ (bounded by polynomial $q(i)$, say) is a function from $f : \mathbb{N} \to \mathbb{N}$ such that 1) $f(n) = p(n)$ and 2) f is computable in polynomial time in n.

3. The realizability set of strings is given by $\mathbf{S}_* = (\{0,1\}^*, \Vdash_{\mathbf{S}_*})$ where

$$i, \ulcorner w \urcorner \Vdash_{\mathbf{S}_*} w \qquad \text{iff} \qquad lh(w) = i$$

4. The realizability set of booleans is given by $\mathbf{B}_* = (\{t, f\}, \Vdash_{\mathbf{B}_*})$ where

$$i, \mathbf{tt} \Vdash_{\mathbf{B}_*} t \qquad \text{and} \qquad i, \mathbf{ff} \Vdash_{\mathbf{B}_*} f$$

   for all natural numbers i.

We have a function $\phi : \{0,1\}^* \to [\![\mathbf{S}]\!]_\rho^{Set}$ defined as follows:

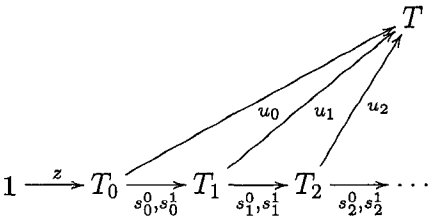
$$\phi(w)_C(f, g, z) = h_{lh(w)}(h_{lh(w)-1} \cdots (h_2(h_1(z))))$$

where $h_i = f$ if the ith digit of w is a 0, and $h_i = g$ if the ith digit of w is a 1. We have function $\psi : [\![\mathbf{B}]\!]_\rho^{Set} \to \{t, f\}$ defined by:

$$\psi(x) = x_{\{t,f\}}(t, f)$$

Our goal next will be to show that the above two mediating functions are in fact realizability morphisms.

Lemma 6.4.1. (Iteration Lemma). *Consider the following diagram of realizability sets and morphisms*



Suppose that the sequence of terms $\{e^0_{s_i}\}_{i\geq 0}$ *is bounded by polynomial* $p_0(i)$, *and* $\{e_{u_i,i}\}_{i\geq 0}$ *is bounded by polynomial* $q(i)$.

Then the function $f : \{0,1\}^* \to T$ *defined by* $f(\epsilon) = u_0(z(*))$ *and* $f(i_n i_{n-1} \cdots i_1) = u_n(s^{i_n}_{n-1}(\cdots s^{i_2}_1(s^{i_1}_0(z(*)))))$ *is a realizability morphism from* $\mathbf{S}_*$ *to* T.

Proof. For each string $w \in \{0,1\}^*$, we have at level $n = lh(w)$,

$$\begin{aligned}
n, e_z &\Vdash_{T_0} z(*)\\
n, e_{s^{i_1}_0} e_z &\Vdash_{T_1} s^{i_1}_0(z(*))\\
&\vdots\\
n, e_{s^{i_n}_{n-1}}(e_{s^{i_{n-1}}_{n-2}} \cdots (e_{s^{i_1}_0} e_z)) &\Vdash_{T_n} s^{i_n}_{n-1}(s^{i_{n-1}}_{n-2} \cdots s^{i_1}_0(z(*)))\\
n, e_{u_n}(e_{s^{i_n}_{n-1}}(e_{s^{i_{n-1}}_{n-2}} \cdots (e_{s^{i_1}_0} e_z))) &\Vdash_T f(w)
\end{aligned}$$

We define terms e_i recursively by

$$\begin{aligned}
e_0 &= \lambda b \otimes r.e_{u_0}(re_z I)\\
e_{i+1} &= \lambda b \otimes r.e_{u_{i+1}}(be_z e_{s^0_i} e_{s^1_i}(\mathbf{ff}(e_i r)))
\end{aligned}$$

For all strings w of length n, we have

$$n, e_n{}^{\ulcorner} w^{\urcorner} \Vdash_T f(w)$$

Finally, the e_n can be bounded by a polynomial in $O(n \cdot p(n) + q(n))$. □

Lemma 6.4.2. *The* ψ *defined above is a realizability morphism from* $[\![\mathbf{B}]\!]^{\mathcal{B}_s}_{\rho}$ *to* $\mathbf{B}_*$.

Proof. Instantiate the second-order variable α with $\mathbf{B}_*$ and afterwards apply it to the pair (t, f). □

The following theorem is a semantic proof of Lafont's result that any generic proof net for the sequent $\mathbf{S}^{(n)} \vdash \mathbf{B}$ defines a polynomial time algorithm that takes a boolean string as input and returns a boolean value as output.

Theorem 6.4.3. *Let π be a generic proof of $\mathbf{S}^{(n)} \vdash \mathbf{B}$ in $ISLL_2$. Let $f : \{0,1\}^* \to \{t, f\}$ be the function:*

$$f(w) = \psi([\![\pi]\!](\underbrace{\phi(w), \cdots, \phi(w)}_{n \text{ times}}))$$

Then: $f(w)$ is computable in polynomial time in $lh(w)$. Moreover, an algorithm for f can be effectively obtained from the proof π.

Proof. First observe that for fixed n, the term $\ulcorner w \urcorner \otimes \cdots \otimes \ulcorner w \urcorner$ is computable in linear time in $lh(w)$. Applying the Iteration Lemma to the denotations of formulas $\mathbf{1}$, $\mathbf{S}\langle 0\rangle$, $\mathbf{S}\langle 1\rangle$, $\mathbf{S}\langle 2\rangle$, ..., $\mathbf{S}$ and the proofs given in the figures below shows that the function $\phi : \{0,1\}^* \to [\![\mathbf{S}]\!]^{Set}$ is a morphism from $\mathbf{S}_*$ to $[\![\mathbf{S}]\!]^{\mathcal{B}^s}$ and so ϕ^n is a realizability morphism from $\mathbf{S}_*^n$ to $[\![\mathbf{S}^{(n)}]\!]^{\mathcal{B}^s}$. It follows that the function $f = \psi \circ [\![\pi]\!]^{Set} \circ \phi^n$ in the theorem is a morphism from $\mathbf{S}_*^n$ to $\mathbf{B}_*$. □

$$\dfrac{\dfrac{\dfrac{\dfrac{\alpha \vdash \alpha}{\mathbf{1}, \alpha \vdash \alpha}\,{\scriptstyle \mathbf{1}L}}{\vdash \mathbf{1} \multimap (\alpha \multimap \alpha)}\,{\scriptstyle \multimap R \text{ 2 times}}}{\vdash \mathbf{S}\langle 0\rangle}\,{\scriptstyle \forall R}}{\mathbf{1} \vdash \mathbf{S}\langle 0\rangle}\,{\scriptstyle \mathbf{1}L}$$

Figure 1: Generic proof of $\mathbf{1} \vdash \mathbf{S}\langle 0\rangle$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{(n)} \vdash ((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{n}} \quad \dfrac{\dfrac{}{\alpha \vdash \alpha} \quad \dfrac{}{\alpha \vdash \alpha}\,{\multimap}L}{\alpha \multimap \alpha, \alpha \vdash \alpha}\,{\multimap}L}{((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{n} \multimap (\alpha \multimap \alpha), ((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{(n)}, \alpha \vdash \alpha}\,\forall L}{\mathbf{S}\langle n\rangle, ((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{(n)}, \alpha \vdash \alpha} \quad \dfrac{}{\alpha \vdash \alpha}\,{\multimap}L}{\mathbf{S}\langle n\rangle, ((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{(n)}, (\alpha \multimap \alpha), \alpha \vdash \alpha}\,\&L1}{\mathbf{S}\langle n\rangle, ((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{(n)}, (\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha), \alpha \vdash \alpha}\,\otimes L\ n{-}1\text{ times}}{\mathbf{S}\langle n\rangle, ((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{n+1}, \alpha \vdash \alpha}\,{\multimap}R\ 2\text{ times}}{\mathbf{S}\langle n\rangle \vdash ((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{n+1} \multimap (\alpha \multimap \alpha)}\,\forall R}{\mathbf{S}\langle n\rangle \vdash \mathbf{S}\langle n+1\rangle}$$

Figure 2: Generic proof of $\mathbf{S}\langle n\rangle \vdash \mathbf{S}\langle n+1\rangle$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{(n)} \vdash ((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{n}} \quad \dfrac{\dfrac{}{\alpha \vdash \alpha} \quad \dfrac{}{\alpha \vdash \alpha}\,{\multimap}L}{\alpha \multimap \alpha, \alpha \vdash \alpha}\,{\multimap}L}{((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{n} \multimap (\alpha \multimap \alpha), ((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{(n)}, \alpha \vdash \alpha}\,\forall L}{\mathbf{S}\langle n\rangle, ((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha))^{(n)}, \alpha \vdash \alpha}\,!L}{\mathbf{S}\langle n\rangle, !((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha)), \alpha \vdash \alpha}\,{\multimap}R\ 2\text{ times}}{\mathbf{S}\langle n\rangle \vdash\ !((\alpha \multimap \alpha)\,\&\,(\alpha \multimap \alpha)) \multimap (\alpha \multimap \alpha)}\,\forall R}{\mathbf{S}\langle n\rangle \vdash \mathbf{S}}$$

Figure 3: Non-generic proof of $\mathbf{S}\langle n\rangle \vdash \mathbf{S}$.

Chapter 7

Stratified Finite MO-Games and FCT

In this chapter, we use the S-construction on a base category of finite discreet games (see [36]) to get a categorical model of IMSAL. We use the term MO-games, after Murawski-Ong, for discreet games. By placing a uniformity condition on sequences of strategies, we show that stratified finite MO-games is fully-complete for IMSAL.

We also consider a model of IMSLL obtained by applying directly the product formula from Chapter 4 in a simple category of AJM-games. It turns out that these games satisfy an ascending chain condition (i.e. there are no infinite increasing chains w.r.t. prefix order of plays in their games trees). Such games may be infinite, but do not have infinite length plays, hence it will be easy to show that total strategies on such games naturally compose (since there can be no 'infinite chattering'). This is in contrast to game models of IMELL, which do not satisfy such a condition. This highlights a natural mathematical (semantic) property that distinguishes between models of predicative copying (i.e. multiplexing) versus models of impredicative copying (i.e. contraction) in linear logic.

Finally, we consider the case of bounding the size and/or depth complexity of sequences of strategies in the S-construction. This will not play a role in full-completeness, but may nevertheless be of interest. We briefly discuss this possibility in the final section of this chapter.

Notation

We shall use the following notation in this chapter. We write $X+Y = \{1\}\times X \cup \{2\}\times Y$ for the disjoint union of sets X, Y. Given functions $f : X \to Z$ and $g : Y \to Z$, we write $[f, g] : X + Y \to Z$ for the canonical map defined by $[f, g](1, x) = f(x)$ and $[f, g](2, y) = g(y)$. Given a set X, we write X^* for the set of finite sequences of elements of X; the empty sequence will be denoted by ϵ. If $Y \subseteq X$ and $s \in X^*$, we write $s \restriction Y$ for the sequence obtained by deleting all elements in $X \backslash Y$ from s. We also write $\mathbb{N}^*$ for $\mathbb{N}\backslash\{0\}$. Further notation will be introduced as needed.

7.1 Games and Strategies

In this section we present an AJM-style game semantics following the presentations in [1] and [36] (see also [4] for a standard reference). The games we consider are two-player games between P (Player) and O (Opponent). Since we are only interested in the intuitionistic (negative) fragment, we shall assume that every play is started by O, and thereafter it alternates between P and O.

A *game* G is a triple $\langle M_G, \lambda_G, P_G \rangle$, where

- M_G is the set of moves
- $\lambda_G : M_G \to \{P, O\}$ is a labeling function designating each move as by Player or Opponent
- P_G is a non-empty, prefix closed subset of M_G^{alt}, the set of finite alternating sequences of moves in M_G (including the empty sequence ϵ), each beginning with an O-move; we call elements of P_G *positions* or *plays*. Thus P_G represents the game tree.

A game is said to be *finite* if both M_G and P_G are finite sets. Henceforth we shall assume that our games are finite, unless otherwise stated.

Some Constructions on Games

The *tensor game* $A \otimes B$ and *linear implication game* $A \multimap B$ are defined as follows. For $\circledcirc \in \{\otimes, \multimap\}$, we have

$$\begin{aligned} M_{A \circledcirc B} &= M_A + M_B \\ P_{A \circledcirc B} &= \{s \in M^{\mathrm{alt}}_{A \circledcirc B} |\ s \upharpoonright A \in P_A, s \upharpoonright B \in P_B\} \end{aligned}$$

with $\lambda_{A \otimes B} = [\lambda_A, \lambda_B]$ and $\lambda_{A \multimap B} = [\overline{\lambda_A}, \lambda_B]$, where $\overline{P} = O$ and $\overline{O} = P$. The *empty game* $\mathbf{1} = \langle \emptyset, \emptyset, \{\epsilon\} \rangle$ is the tensor unit, and as we shall see later it is also a terminal object in the category of games.

The following are consequences of the definitions [1]. *O-Switching Condition*: in any play $s \in P_{A \otimes B}$, if successive moves $s_j s_{j+1}$ are in different components (i.e. one is from A the other from B), then s_j is a P-move and s_{j+1} is an O-move. In other words, only O can switch from one subgame to another. Similarly it follows that every play $s \in P_{A \multimap B}$ satisfies the *P-Switching Condition*: if two consecutive moves in s are in different components, then the first is an O-move and the second a P-move, i.e. only P can switch components.

For each natural number n, we define the (bounded) exponential game $!_n A$ by:

$$\begin{aligned} M_{!_n A} &= M_A \times \{\star, 1, \ldots, n\} \\ \lambda_{!_n A}(m, i) &= \lambda_A(m) \\ P_{!_n A} &= \{s \in M^{\mathrm{alt}}_{!_n A} \mid \forall i \in \{\star, 1, \ldots, n\}. s \upharpoonright i \in P_A\} \end{aligned}$$

The index $\star$ is mainly used to identify when a !-formula comes from an exponential box (i.e. soft promotion), and is used to define the network protocol on IMSAL. (See Remark 7.1.3 for another illustration of its use.)

Strategies and Categories of Games

The following definitions are standard and taken from [36]. A *deterministic P-strategy*, or simply *strategy*, for a game G is a non-empty, prefix-closed subset σ of P_G satisfying: (i) (contingent completeness) for any even-length s, if $s \in \sigma$ and $sm \in P_G$, then $sm \in \sigma$, and (ii) (determinacy) if even-length sm and sm' are both in σ, then $m = m'$.

A strategy σ on G is *history-free* if there is a partial function $f : M_G^O \rightharpoonup M_G^P$ such that for any odd-length $sm \in \sigma$, we have $smm' \in \sigma$ iff $f(m)$ is defined and equal to m'. We write $\sigma = \sigma_f$ just in case f is the least such function. Further it is said to be *injective history-free* if the least such f is injective. If for every odd-length $s \in \sigma$, there is some m such that $sm \in \sigma$, we say that σ is *total*. For any games A_1, A_2, A_3 we define $\mathcal{L}(A_1, A_2, A_3)$ to be the set of finite sequences s of moves from $M_{A_1} + M_{A_2} + M_{A_3}$ such that for any pair of consecutive moves $mm' \in s$, if $m \in M_{A_i}$ and $m' \in M_{A_j}$ then $|i - j| \leq 1$. (We call $\mathcal{L}(A_1, A_2, A_3)$ the set of *interaction sequences* over (A_1, A_2, A_3).) Take strategies σ and τ for games $A \multimap B$ and $B \multimap C$ respectively. Then composition is defined by:

$$\sigma;\tau \;=\; \{s \upharpoonright (A,C) \mid s \in \mathcal{L}(A,B,C) \wedge s \upharpoonright (A,B) \in \sigma \wedge s \upharpoonright (B,C) \in \tau\}$$

Games and strategies (maps from A to B are strategies in $A \multimap B$) form a symmetric monoidal closed category; the category whose maps are injective history-free total (finiteness is necessary here) strategies form a subcategory [1].

Interpreting IMSAL$_f$ formulas

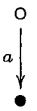
As in [36], it will be convenient to fix representations of moves in the monoidal structure as follows:

$$\begin{aligned} M_{A \otimes B} &= M_A \times \{l\} \cup M_B \times \{r\} \\ M_{A \multimap B} &= M_A \times \{L\} \cup M_B \times \{R\} \end{aligned}$$

Formulas are interpreted inductively as games as follows:

$$\begin{aligned} [\![\mathbf{1}]\!] &= \langle \emptyset, \emptyset, \{\epsilon\} \rangle \\ [\![a]\!] &= \langle \{a\}, \{(a, O)\}, \{\epsilon, a\} \rangle \qquad (a \text{ is any atom}) \\ [\![A \otimes B]\!] &= [\![A]\!] \otimes [\![B]\!] \\ [\![A \multimap B]\!] &= [\![A]\!] \multimap [\![B]\!] \\ [\![!_n A]\!] &= !_n [\![A]\!] \end{aligned}$$

where on the right we are using the corresponding game constructions. So each atomic formula a is interpreted as a single-move atomic game with game tree



In the game context, a is called a *token*. In the following we shall often abuse the notation and equate formulas with their corresponding game interpretations (i.e. we usually omit the semantic brackets $[\![-]\!]$ when the context is clear). From now on, by game we shall mean a game which is the interpretation of some IMSAL_f formula. A move of a game has the form

$$((\cdots((a, i_n), i_{n-1}\cdots), i_1)$$

where a is a token and each $i_j \in \mathbb{N}^* \cup \{\star, l, r, L, R\}$. We may therefore consider a move as a pair $(a, i_n \cdots i_1)$, where a is a token and where the second component is called its *occurrence*. We have two notions of "depth of a move", both of which will be important. The *degree of a move* is defined as the total number of $\star$s in its occurrence, and the *depth of a move* is defined as the total number of $\star$s and numbers in its occurrence. (So the degree of a move is always less than of equal to the depth of a move.) The *index at depth* j or j*-index* ($j \geq 1$) of a move is defined as the depth j subsequence of its occurrence of the form $i_{r_j} i_{r_j - 1} \cdots i_3 i_2 i_1$ such that $i_{r_j} \in \{\star\} \cup \mathbb{N}^*$. For example, the 2-index of the move $(a, 2r \star l8L)$ is $\star l8L$. The *depth of a game* G is defined to be the maximum depth of its moves. Finally, a strategy σ for a game is said to be *token-reflecting* if for any even-length $smm' \in \sigma$, m and m' have the same token.

We call token-reflecting, injective history-free, total strategies IMAL-winning. Indeed, these properties are enough to get a fully-complete model of the $(\otimes, \multimap)$-fragment of intuitionistic multiplicative affine logic, which we abbreviate to IMAL. See [36] and references therein for a proof of the following.

Theorem 7.1.1. (IMAL Full Completeness [36]). *For any game given by an IMAL-sequent $\Gamma \vdash A$, and for any IMAL-winning strategy σ for the game, there is a derivation of $\Gamma \vdash A$ whose derivation is σ.*

$\approx$-reflexive games

In order to get a finitary multiplexor category, we shall require a standard notion of partial equivalence relation $\approx$ over strategies (see [4, 36]). A *$\approx$-game* G is defined to be a four-tuple $\langle M_G, \lambda_G, P_G, \approx_G \rangle$ where $\langle M_G, \lambda_G, P_G \rangle$ is a game, and $\approx_G$ is an equivalence relation on P_G satisfying:

- $sa \approx_G tb \quad \Rightarrow \quad s \approx_G t$
- $s \approx_G t \quad \Rightarrow \quad \lambda^*_G(s) = \lambda^*_G(t)$
- $s \approx_G t \wedge sa \in P_G \quad \Rightarrow \quad \exists b \in M_G.tb \in P_G \wedge sa \approx_G tb.$

The game constructions defined earlier can be extended to their respective $\approx$-game constructions as follows. For $\circledcirc \in \{\otimes, \multimap\}$, we have $s \approx_{A \circledcirc B} t$ just in case $s \restriction A \approx_A t \restriction A$, $s \restriction B \approx_B t \restriction B$ and $\pi^*_2(s) = \pi^*_2(t)$. For the exponential games $!_n A$, we define $s \approx_{!_n A} t$ iff

$$\exists \alpha \in S^*(n). \forall i \in \{\star, 1, \ldots, n\}. s \restriction i \approx_A t \restriction \alpha(i) \ \wedge \ \alpha^*(\pi^*_2(s)) = \pi^*_2(t)$$

where $S^*(n)$ is the set of permutations α on $\{\star, 1, \ldots, n\}$ such that $\alpha(\star) = \star$. The equivalence relation $\approx_G$ extends to a partial equivalence relation (i.e. symmetric and transitive) over strategies of G: $\sigma \approx_G \tau$ holds just in case for all $s \in \sigma$, $t \in \tau$, $sa \in P_G$, $tc \in P_G$, if $sa \approx_G tc$ then the following properties hold:

- $sab \in \sigma \quad \Rightarrow \quad \exists d \in M_G.tcd \in \tau \wedge sab \approx_G tcd$
- $tcd \in \tau \quad \Rightarrow \quad \exists b \in M_G.sab \in \sigma \wedge sab \approx_G tcd.$

A strategy σ is said to be *$\approx$-reflexive* if $\sigma \approx \sigma$, and in this case we write $[\sigma] = \{\tau | \ \sigma \approx \tau\}$. It turns out that $\approx$ is preserved by composition, so that games, with maps from A to B given by token-reflecting, injective history-free, $\approx$-reflexive, total strategies in $A \multimap B$ form a symmetric monoidal closed category (see [4]). Moreover, we have the following result:

Proposition 7.1.2. *The above described category is a finitary affine multiplexor category.*

Proof. It is straightforward to check that the following partial functions define token-reflecting, injective history-free, $\approx$-reflexive, total strategies:

- *functoriality:* let f define a token-reflecting, injective history-free, $\approx$-reflexive, total strategy in $A \multimap B$. Then the strategy in $!_nA \multimap !_nB$ defined by partial function $!_nf$ is described as follows. Suppose $f(m, \delta) = (m', \delta')$ where $\delta, \delta' \in \{L, R\}$, we define

$$!_nf : ((m, i), \delta) \mapsto ((m', i), \delta')$$

  where $i \in \{\star, 1, \ldots, n\}$.

- *monoidality:* a) for each natural number n, we define a strategy in $!_nA \otimes !_nB \multimap !_n(A \otimes B)$ by the following partial function:

$$b_{2,n} : \begin{cases} (((m, \delta), j), R) & \mapsto \ (((m, j), \delta), L) \\ (((m, j), \delta), L) & \mapsto \ (((m, \delta), j), R) \end{cases}$$

  where $\delta \in \{l, r\}$ and $j \in \{\star, 1, \ldots, n\}$; b) $\mathbf{1} = !_n\mathbf{1}$, so $b_{0,n}$ on $\mathbf{1} \multimap !_n\mathbf{1}$ is just the identity on $\mathbf{1}$ (i.e. the empty strategy ϵ).

- *multiplexing of rank* (i, n)*:* let $\alpha : \{1, \ldots, n\} \to \{1, \ldots, i\}$ be an idempotent injection. We define a strategy on $!_iA \multimap A_1 \otimes \cdots \otimes A_n$ by the following partial function:

$$(\mu_{i,n}, \alpha) : ((m, j), \delta) \mapsto ((m, \alpha(j)), \delta') \quad \text{where } \delta, \delta' \in \{L, R\} \text{ and } \delta \neq \delta'$$

  where $j \in \{\star, 1, \ldots, n\}$. For ease of reference, we write $A_1, \cdots, A_n$ to index the n occurrences of A in A^n, and a move m in A_j is written as the pair (m, j). Observe that $(\mu_{i,n}, \alpha) \approx (\mu_{i,n}, \beta)$ for any idempotent injection $\beta : \{1, \ldots, n\} \to \{1, \ldots, i\}$.

Finally, the coherence diagrams follow easily via the symmetry built into the equivalence relation on positions. $\square$

Remark 7.1.3. Observe that the above strategies are degree-preserving. When we introduce the network protocol for IMSAL, it will be a consequence that strategies are degree-preserving. Degree-preserving strategies have some interesting properties. E.g. one can refute the IMSAL$_f$ formula $!_{x+y}A \multimap !_xA \otimes !_yA$, where x and y are natural numbers: e.g. let $x = y = 1$ and let A be the atomic game $\langle \{a\}, \{(a, O)\}, \{\epsilon, a\}\rangle$. Then to be degree-preserving we must have $(a, \star lR) \mapsto (a, \star L)$ and $(a, \star rR) \mapsto (a, \star L)$, which is not injective. As a second example, there is no degree-preserving (dereliction) strategy in $!_0A \multimap A$: again take A to be atomic, then the opening move has no response. However, there is a degree-preserving (copy-cat) strategy in $!_xA \multimap !_yA$, for $x \geq y$. This can be avoided by giving a different "$\star$" for each natural number, and requiring strategies to be preserve the number of $\star$s of each type.

Relative Completeness

Let σ_f be a degree-preserving, history-free strategy on a game G defined by partial function $f : M_G^O \rightharpoonup M_G^P$. We say that σ_f is *stable* if for all $x \in \{1, \ldots, n\}$ and for any degree i the following diagram commutes:

$$\begin{array}{ccc} M_G^O & \xrightarrow{f} & M_G^P \\ {\scriptstyle [x/\star]_i}\downarrow & & \downarrow{\scriptstyle [x/\star]_i} \\ M_G^O & \xrightarrow{f} & M_G^P \end{array}$$

where $[x/\star]_i$ denotes the operation on moves of replacing $\star$ with x at degree i in the occurrence. It is not hard to show that stable strategies compose.

For any natural number n, let IMSAL^n denote IMSAL$_f$ restricted to the single exponential $!_n$. A formula in IMSAL is interpreted by the S-construction as a sequence of games $G_0, G_1, G_2, \ldots$, where for each n, G_n is the game interpretation of a formula in IMSAL^n. Given a strategy σ in G_n there is a canonical extension to a strategy σ' in G_{n+1}. More precisely, suppose σ is defined by partial function f and let $\star_2 = n+1$. We define a strategy σ' by the partial function f', where $m \in \mathrm{dom}(f')$ iff $m[\star/\star_2] \in \mathrm{dom}(f)$. (This notation means replace all $\star_2$s in the occurrence of m by $\star$, if there are any.) And moreover, $f'(m) = f(m[\star/\star_2])[\star_2/\star]$, where the latter substitution must be done only at degrees where $\star_2$ occurs in m. We write $\sigma_n \sqsubseteq \sigma_{n+1}$ if $\sigma'_n \approx_{G_{n+1}} \sigma_{n+1}$.

Finally, we say that a sequence of strategies $\sigma_n, \sigma_{n+1}, \ldots$ in games $G_n, G_{n+1}, \ldots$, respectively, is *uniform* if

$$\sigma_n \sqsubseteq \approx \sigma_{n+1} \sqsubseteq \approx \cdots$$

In fact, σ_{i+1} is the smallest possible extension of σ_i (w.r.t. inclusion). Thus uniformity gives the tightest possible connection between levels of strategies, which will be useful for proving full completeness.

Proposition 7.1.4. *Uniform strategies compose.*

Proof. We must show $(\sigma_i; \tau_i)' = \sigma_i'; \tau_i'$. Suppose σ_i is represented by partial function f and τ_i by partial function g. Then $m \in \mathrm{dom}(f'; g')$ iff $m \in \mathrm{dom}(f')$ and $f'(m) \in \mathrm{dom}(g')$ iff $m[\star/\star_2] \in \mathrm{dom}(f)$ and $f'(m)[\star/\star_2] \in \mathrm{dom}(g)$. But $f'(m) = f(m[\star/\star_2])[\star_2/\star]$, so we have $m \in \mathrm{dom}(f'; g')$ iff $m[\star/\star_2] \in \mathrm{dom}(f)$ and $f(m[\star/\star_2]) \in \mathrm{dom}(g)$ iff $m[\star/\star_2] \in \mathrm{dom}(f; g)$ iff $m \in \mathrm{dom}(f; g)'$. Moreover, we have the equalities $f'; g'(m) = g'(f'(m)) = g'(f(m[\star/\star_2])[\star_2/\star]) = g(f(m[\star/\star_2]))[\star_2/\star] = (f; g)'(m)$. □

We model IMSAL with stratified finite games and IMSAL-winning (i.e. uniform sequences of stable, networked (see next section), token-reflecting, injective history-free, $\approx$-reflexive, total) strategies by applying the S-construction on finite games. We have the following result.

Proposition 7.1.5. (Relative Completeness.) *If $\sigma = (k : \sigma_i)$ is an IMSAL-winning strategy in a sequence of games $G_0, G_1, \ldots$, given by an IMSAL sequent, such that for some level n, σ_n is the denotation of a proof π_n in IMSAL^n, then in fact σ is the denotation of the corresponding IMSAL proof π_n^*.*

Proof. We need to show for each $i \geq n$, that σ_i is the denotation of proof π_i, the proof obtained by replacing the index n by i in π_n. By induction on $i - n$, we have $\sigma_{i+1} = \sigma_i' = [\![\pi_i]\!]' = [\![\pi_{i+1}]\!]$, where the first equality follows by uniformity, the second by the induction hypothesis, and the third by soundness of the interpretation. □

Therefore we have effectively reduced the full-completeness problem for IMSAL to the finitary version at fixed level n.

7.2 Network Protocol for IMSAL^n

In [36], the authors give a semantic analysis of the IMLAL exponential connectives ! and §. By organizing threads into networks in accord with a protocol and demanding that strategies comply with it, the authors prove a full-completeness result for IMLAL. They formalize all this with the notion of a *network function* and strategies that comply with the protocol are called *discreet*. In this section, we introduce a suitable modification of this technique for handling the exponential boxes in IMSAL^n, which may have applications to other systems of linear logic. In this section, we shall only be interested in games G_n which are the interpretations of formulas in IMSAL^n, for some fixed natural number n.

A *thread of* G_n is named by an i-index θ of G_n, and is defined as the set of moves of G_n whose i-index is θ. If there are an odd number of occurrences of L in θ, then the thread named by θ is called a P-thread, otherwise it is called an O-thread. We write T_{G_n} for the set of threads of a game G_n, which partitions into $T^O_{G_n}$ (the set of O-threads) and $T^P_{G_n}$ (the set of P-threads) [36]. Finally, a thread named by θ is called a $\star$-*thread* if the leftmost symbol in θ is a $\star$. Since IMSAL-strategies are not depth-preserving, we shall not consider threads at a fixed depth i, as in [36]. The appropriate notion for us will be degree-preservation.

A *network function* is a partial function of the form:

$$\eta_{G_n} : T^P_{G_n} \rightharpoonup T^O_{G_n}$$

We say a thread is *boxed* if either it is an O-thread, a $\star$-thread, or is in $\mathrm{dom}(\eta_{G_n})$. We say a boxed thread has *box-depth* i if it has a total of i boxed subthreads. We assume the following additional properties on network functions:

1. η_{G_n} is box-depth preserving.

2. if $\eta_{G_n}(t_1) = t_2$ is defined, then $\eta_{G_n}(t_1') = t_2'$, where t_1' is obtained from t_1 by replacing numbers with $\star$s in all boxed positions, and similarly for t_2'.

3. if $\eta_{G_n}(t_2 \star t_1) = s_2 \star s_1$ (with $\star$ at box-depth i), then $\eta_{G_n}(t_2 x t_1) = s_2 x s_1$ for all $x \in \{\star, 1, \ldots, n\}$. Moreover, $\star t_1$ and $\star s_1$ belong to the same network, i.e. one of the four properties below holds.

Note that it is a consequence of the definition that network functions are degree-preserving. A *thread function* (at box-depth i) is a partial function of the form:

$$t_{G_n,i} : M_{G_n} \rightharpoonup T_{G_n}$$

that maps a move to the box-depth-i thread in which it occurs.

We say η_{G_n} *networks* a position $s \in P_{G_n}$ of even length at box-depth i just in case for each odd j, $t_{G_n,i}(s_j)$ is defined iff $t_{G_n,i}(s_{j+1})$ is, and if both are defined, one of the following holds:

$$\begin{array}{llll} (i) & \eta(t(s_j)) = t(s_{j+1}) & (ii) & t(s_j) = \eta(t(s_{j+1})) \\ (iii) & \eta(t(s_j)) = \eta(t(s_{j+1})) & (iv) & t(s_j) = t(s_{j+1}) \end{array}$$

We say a strategy σ_n in G_n is *networked* if there exists a network function η_{G_n} that networked every $s \in \sigma_n$ at every box-depth up to the depth of G_n. Observe that networked strategies are necessarily degree-preserving. To show directly that networked strategies compose is delicate. (See [36] for a proof from first principles for IMLAL.) Instead, we shall obtain this result as a corollary to full completeness.

We call a strategy *generic* if it can be networked by a *total* network function η_{G_n}. In this case, the box-depth is the same as the depth and hence the network function and strategy are both depth- and degree-preserving. Then the network function η_{G_n} can be partitioned into a family of network functions $\eta_{G_n,i}$, one for each depth i up to the depth of G_n. In this case, our notion of network function can be viewed as a network function in the sense of Murawski and Ong's [36].

7.3 IMSAL Full Completeness

In this section we prove that stratified finite games with IMSAL-winning strategies is a fully complete model of IMSAL. This will follow by relative completeness as a corollary to the following theorem.

Theorem 7.3.1. (Full Completeness at Level n). *For any winning strategy σ for the game G_n given by an IMSAL^n sequent $\Gamma \vdash C$, there is a derivation Ξ of the sequent such that σ is the denotation of Ξ.*

The proof of this theorem in the special case that σ is generic can be extracted from the full-completeness proof for IMLAL by Murawski and Ong in [36], which we shall do next. Hence the following proof (in the generic case) is only a slight modification to the one given in [36]. We hope to give a more "first principles" proof of the above theorem another time. Unless otherwise stated, we assume n is fixed and for ease of writing, we will usually suppress the index in the formulas and strategies. We begin with the following lemmas.

Lemma 7.3.2. (Deboxing). *For any generic map $F : !A_1 \otimes \cdots \otimes !A_m \to !B$, there is a unique map $f : A_1 \otimes \cdots \otimes A_m \to B$ such that $F = b^*; !f$, where b^* is the canonical map $b^* : !A_1 \otimes \cdots \otimes !A_m \to !(A_1 \otimes \cdots \otimes A_m)$.*

Proof. Since F is generic, it is networked by a depth- and degree-preserving network function. Therefore, any play begun with a move of the form $(m, \star R)$ must have all its moves of the form $(m, \star w)$, where w is either R or the "address" of $!A_i$ for some $1 \leq i \leq m$. This defines the strategy f. Then the fact that $F = b^*; !f$ follows by the stability of F. □

Let σ be a IMAL-winning strategy for a game given by an IMAL-sequent $\Delta \vdash P \otimes Q$ where $\Delta = D_1, \ldots, D_n$. We say that σ is *splitting* just in case there is a partition of Δ into (Δ_1, Δ_2) and IMAL-winning strategies σ_1 and σ_2 for $\Delta_1 \vdash P$ and $\Delta_2 \vdash Q$ respectively such that $\sigma = \sigma_1 \otimes \sigma_2$.

Lemma 7.3.3. (Pivot [36]). *If σ is not splitting, then there is some $D_i = A \multimap B$ in Δ (which we shall call a pivot) and IMAL-winning strategies τ and ν for $\Theta \vdash A$ and $B, \Xi \vdash P \otimes Q$ respectively, where (Θ, Ξ) is a partition of $\Delta \backslash \{D_i\}$, such that σ can be defined in terms of τ and ν.*

Lemma 7.3.4. *Let σ be a winning strategy in game $G_n = A \multimap B$ (at fixed level n). Then σ has a factorization $A \xrightarrow{p} A_1 \xrightarrow{\sigma'} B_1 \xrightarrow{q} B$, where σ' is generic and p and q are canonical.*

Proof. We prove the lemma by induction on the weight of $A \multimap B$. (E.g. $wt(!_n A) = n \cdot wt(A) + 1$, etc.) Let $!C$ be a least depth P-subformula of A whose corresponding

threads are not in $\mathrm{dom}(\eta_{G_n})$, and let $A_1 = A[C^n/!C]$. (Observe that the $\star$-thread of $!C$ is not σ-reachable, because otherwise all $!C$-threads would be in $\mathrm{dom}(\eta_{G_n})$.) Similarly, let $!D$ be a least depth P-subformula of B whose corresponding threads are not in $\mathrm{dom}(\eta_{G_n})$, and let $B_1 = B[D^n/D]$. Using multiplexing, we get a factorization $A \xrightarrow{p} A_1 \xrightarrow{\sigma'} B_1 \xrightarrow{q} B$. The game $A_1 \multimap B_1$ has strictly less weight, and hence we invoke the induction hypothesis to finish the proof. □

Proof of Main Theorem

The following argument (in the generic case) is due to Murawski and Ong [36]. First suppose σ is generic. We prove the theorem in the generic case by induction on the size of the sequent $\Gamma \vdash C$ that defines the game G_n. W.l.o.g. we may assume that no formula in $\Gamma = C_1, \ldots, C_n$ is a tensor, and that every C_i is σ-reachable in the sense that there is a play in σ that contains some C_i-move.

The formula C must have one of the following forms:

1. $C = P \otimes Q$
2. $C = !P$
3. $C = a$, an atom.

Case 1: if σ happens to be splitting (i.e. $\sigma = \sigma_1 \otimes \sigma_2$), then split it and apply the induction hypothesis. Otherwise, we argue that there must be a pivot in Γ. With the pivot in Γ, we obtain two smaller instances and apply the induction hypothesis, finishing case 1. Suppose σ is not splitting. We show that Γ has a pivot as follows. First transform the sequent by adding a free atom as "$x \multimap -$" inside each !-formula from Γ and adding a copy of "$x \multimap -$" of negative polarity inside the O-exponential formula of the corresponding network. E.g. the sequent $!c, !(c \multimap d), !(d \multimap e), !e \multimap !a \otimes !b, !c \vdash !(c \otimes a) \otimes !b$ is transformed to

$$!(x \multimap c), !(y \multimap (c \multimap d)), !(z \multimap (d \multimap e)),$$
$$!(x \multimap (y \multimap (z \multimap e))) \multimap !a \otimes !b, !(v \multimap c) \;\vdash\; !(v \multimap c \otimes a) \otimes !b$$

Call the new sequent $\Gamma' \vdash C'$ and the corresponding strategy σ'. Next we transform the sequent a second time to $\Gamma'' \vdash C''$ by "forgetting all the boxes" and turning it into an IMAL-game, and call the corresponding strategy σ''. The first transformation guarantees that all formulas in Γ'' are $\multimap$-formulas or atoms. So if σ'' is splitting, then so is σ' and hence so is σ. Since σ is not splitting by assumption, we may assume that σ'' is not splitting either. Then by lemma 7.3.3 there must be a pivot in Γ''. This pivot formula cannot be one of the $\multimap$-formulas introduced by the first transformation, since they communicate with the O-thread of their respective networks. So that pivot is also a pivot for $\Gamma \vdash P \otimes Q$.

Case 2: if Γ does not contain a $\multimap$-formula, then every formula in it must be a !-formula and so we may use lemma 7.3.2 to obtain a smaller instance, and then apply the induction hypothesis. Otherwise, $!P$ is part of some network $!G_1, \ldots, !G_l$. For each $G = P, G_1, \ldots, G_l$, we replace $!G$ by $!G \otimes !G$. E.g. the sequent $c, c \multimap !b, !(b \multimap a) \vdash !a$ is transformed to

$$c, c \multimap (!b \otimes !b), !(b \multimap a) \otimes !(b \multimap a) \vdash !a \otimes !a$$

Let σ' be the corresponding strategy for the transformed game $\Gamma' \vdash !P \otimes !P$; σ' is not splitting, because the $\multimap$-formula in Γ' is reachable from both $!P$ on the right. Arguing as in the previous case, there must be a pivot in Γ', which gives a pivot in Γ. With this pivot, we obtain two smaller instances, and we apply the induction hypothesis.

Case 3: if Γ does not contain a $\multimap$-formula, then $\Gamma = a$, and σ is the identity on a. Otherwise, the atom a must occur as a subformula of some formula in Γ (where the response to Opponent's opening move lives). We replace both occurrences of a by $a \otimes a$. E.g. the sequent $b, b \multimap a \vdash a$ is transformed to

$$b, b \multimap a \otimes a \vdash a \otimes a$$

Let σ' be the corresponding strategy for the transformed game $\Gamma' \vdash a \otimes a$; σ' is not splitting, because the $\multimap$-formula in Γ' is reachable from both a on the right. Arguing

as in the case 1, there must be a pivot in Γ', which gives a pivot in Γ. With this pivot, we obtain two smaller instances, and we apply the induction hypothesis.

This completes the proof for generic strategies.

Now suppose that σ is a winning strategy as in the statement of the theorem. By lemma 7.3.4, we have a factorization $\sigma_n = p; \sigma'; q$, where σ' is generic, and p and q are canonical. Hence each is the denotation of an IMSAL^n proof. Using the cut rule we get an IMSAL^n proof of the sequent $\Gamma \vdash C$ whose denotation is σ. Finally, since the denotation is preserved under cut-elimination, we see that σ is in fact the denotation of a cut-free IMSAL^n proof. This completes the proof of the theorem in the general case. □

IMSAL full-completeness now follows immediately by the previous theorem and relative completeness (Prop. 7.1.5). We have therefore proved the main result of this chapter.

Theorem 7.3.5. (IMSAL Full Completeness). *For any IMSAL-winning strategy $\sigma = (n : \sigma_i)$ in a sequence of games $G_0, G_1, \ldots$, given by an IMSAL sequent $\Gamma \vdash C$, there is a derivation Ξ of the sequent such that σ is the denotation of Ξ.*

Corollary 7.3.6. *Networked strategies compose.*

Proof. Let σ and τ be networked strategies. By full-completeness, $\sigma = [\![\pi_1]\!]$ and $\tau = [\![\pi_2]\!]$, for some IMSAL proofs π_1 and π_2. Let π be the cut-free proof obtained by cutting π_1 and π_2 and then eliminating all cuts from the proof. Then $\sigma; \tau = [\![\pi]\!]$, hence it must be networked. □

7.4 Noetherian Games

There is an interesting class of infinite games, which includes the finite ones, that can be used to model IMSLL. To the best of my knowledge, this class of games has not be investigated before. Given a game A, define the set P_A^∞ of infinite plays over A as follows:

$$P_A^\infty = \{s \in M_A^\omega | \mathrm{Pref}(s) \subseteq P_A\}$$

where $\mathrm{Pref}(s)$ denotes the set of finite prefixes of s. We say that a game A is *noetherian* if $P_A^\infty = \emptyset$. Note that such games need not be finite, and may have an infinite number of moves. (But there are no infinite ascending chains of moves in P_A w.r.t. the prefix ordering.) In general games, total strategies do not compose because of 'infinite chattering', and one must use a specification structure in order to ensure that they do [1]. In noetherian games, however, infinite chattering cannot occur and total strategies naturally compose. (In fact, one can see the condition $P_A^\infty = \emptyset$ as a trivial specification structure.) So noetherian games with total, history-free strategies form an autonomous category and a model of IMLL. Moreover, we can model IMSLL with noetherian games using product formula 1 (first without the symmetrized tensor) as follows. Let $F = \{(i,j) | i,j \in \mathbb{N}^*, i \geq j\}$ and F_i denote the subset of F consisting of elements with first index i. Then we define:

$$\begin{aligned} M_{!A} &= M_A \times F \\ \lambda_{!A}(a,f) &= \lambda_A(a) \\ P_{!A} &= \{s \in M_{!A}^{\mathrm{alt}} | \exists i \in \mathbb{N}^*.(\forall f \in F_i.s \restriction f \in P_A \wedge \forall f \in F \backslash F_i.s \restriction f = \epsilon)\} \end{aligned}$$

where $s \restriction (i,j)$ stands for the projection of s on the (i,j)-th copy of A. Observe that if A is noetherian, then so is $!A$. Finally, by incorporating an appropriate equivalence relation on plays, we find that noetherian games with total, history-free strategies form a multiplexor category and a model of IMSLL. Our attempts at using product formula 2 directly to model IMSAL always led to non-history-free strategies. This problem is avoided by using the S-construction.

In linear logic, the exponential game $!A$ is defined as an infinite symmetrized tensor product $A^{\otimes_s \omega}$ [36]. This construction is not allowed in noetherian games. Indeed, the contraction and digging strategies have infinite-length plays (even the identity strategy can have infinite length plays in this game). Therefore noetherian games do not model IMELL (at least with the usual interpretation). As another application, the "infinity numeral" strategy on the (first-order) Church numerals game $!(a \multimap a) \multimap (a \multimap a)$ [36] is also ruled out in noetherian games. Therefore, noetherian games are a natural extension of finite games, which model IMSLL but not IMELL. The semantic ACC (ascending chain condition) property in noetherian games provides

a pleasing mathematical explanation for the difference between impredicative and predicative copying in linear logic.

7.5 Bounded Complexity Games

It might be interesting to restrict the (size- or depth-) complexity of sequences of strategies in the S-construction. In other words, a morphism is a sequence of strategies $\sigma_n \subseteq \sigma_{n+1} \subseteq \cdots$, as usual, but we now require that there exists a polynomial P such that the length of any play in σ_i is $\leq P(i)$. This is similar to what we did in our realizability semantics in chapter 6, by bounding sequences of terms to polynomial size. It corresponds to our intuition of IMSAL proofs as circuit families with polynomial depth complexity.

Chapter 8

Conclusions and Further Work

We have justified the design of soft linear logic by showing that it possesses a very natural mathematical interpretation. A key point we have shown is that there are natural mathematical properties that distinguish between models of linear logic with predicative copying (i.e. multiplexing) versus models of linear logic with impredicative copying (i.e. contraction). The noetherian property for games is one such example. Moreover, models of linear logic with predicative copying allow for a stratification into *finite* approximations, which is a crucial point for our analogy to uniformly polynomial circuits. These results are an important first step towards a truly semantical understanding of polynomial time complexity.

We conclude with some questions and possible extensions to this work:

- It would be interesting to explore in greater detail the relationship between PCL and soft linear logic. Although the two system are certainly related, it is not so obvious what the precise relationship is, since the latter is typed while the former is not. One direction would be to introduce a typed system of PCL in which Abramsky's encoding of PTIME algorithms is typable. Presumably, this would lead to a subsystem of soft linear logic. The other direction would be to avoid types altogether and try to understand what a categorical model of PCL is. With this approach, one could first formalize what is means to be a model of affine combinatory logic (which we may call, *affine Turing categories*, after [14]), and then use ideas from this thesis (i.e. the S-construction) to construct

models of PCL. This approach is also appealing because one may be able to make some connections with the characterizations of PTIME within the lambda calculus (see e.g. [7, 31]).

- An obvious direction would be to extend our game semantics to the second-order system and prove a full completeness result there (Abramsky explains how to handle polymorphism in games in [1].) Another is to investigate whether our game model is *faithfully* complete with respect to the syntactic category of IMSAL. One could also look into other interpretations, for example, in Ehrhard's category of hypercoherences. Perhaps even a functorial semantics (see [9] and references therein).

- A long standing open question in linear logic is whether provability in MELL is decidable. What about MSLL with its restricted exponentials? We conjecture that the two problems are equivalent. A first step at proving this conjecture was made in [35], by attempting to prove the completeness of MELL with respect to a semi-linear phase semantics. With this result, decidability of MELL follows as a corollary. This novel approach did not succeed in proving the conjecture, but nevertheless it may be instructive to develop a semi-linear phase semantics for MSLL, and for other light linear logics.

- There are many characterizations of PTIME computation, but unfortunately, it is not clear how they are related. In particular, the various systems of light linear logic which characterize PTIME use very different logical principles which seem difficult to unify. It would be informative to organize these logical systems into a common categorical framework. Perhaps then we can identify the key structural properties which characterize PTIME computation. For example, is a notion of stratification somehow inherent in bounded time computation? At this point, we do not know.

- Finally, recent interest in using linear logic as a basis for understanding quantum computation leads one to wonder whether the light exponentials have any physical interpretation. For example, in view of the "no-cloning" axiom in quantum

computation, we wonder if there is any quantum analog to multiplexing.

Appendix A

Proof of Cut-Elimination for IMSLL

This proof of cut-elimination is adapted from Lincoln et al [32]. We define the weight of a IMSLL formula A to be its size, i.e.

$$\begin{aligned} w(A \circledcirc B) &= w(A) + w(B) + 1, \quad \circledcirc \in \{\otimes, \multimap\} \\ w(!A) &= w(A) + 1 \\ w(\alpha) &= 1 \\ w(\mathbf{1}) &= 1 \end{aligned}$$

For technical reasons, we will eliminate the following (derived) cut rule:

$$\frac{\Gamma \vdash A \qquad \Delta_1, A, \Delta_2 \vdash C}{\Gamma, \Delta_1, \Delta_2 \vdash C}\text{cut}$$

In this case, we call A the *cut-formula.* (Note that this rule corresponds to the old cut rule when $\Delta_2 = \emptyset$.) The weight of a proof is defined to be maximum weight of any cut-formula in the proof, or zero if the proof is cut-free.

The *principal formula* of an application of an inference rule is defined to be any formula which is introduced by that rule.

Lemma A.0.1. *Given a proof of the sequent $\Gamma \vdash C$ in IMSLL which ends in an application of cut with cut-formula weight $d > 0$, and where the weight of the proofs*

of both hypotheses is less than d, we may construct a proof of $\Gamma \vdash C$ *in IMSLL of weight less than d.*

Proof. By induction on the number of proof rules applied in the derivation of $\Gamma \vdash C$.

Given a derivation which ends in a cut, we perform case analysis on the rules which were applied immediately above the cut rule. One of the following cases must apply to any such derivation:

(1) the cut-formula is not principal in one or both hypotheses;

(2) the cut-formula is principal in both hypotheses.

Cut of non-principal formulas

In each of the following cases, the weight of the proof is the same before and after the reduction. However, the last cut rule is pushed up the proof allowing for an application of the induction hypothesis.

Cut (case 1)

$$\dfrac{\Gamma \vdash C \qquad \dfrac{\Delta_1, C, \Delta_2 \vdash A \qquad \Delta_3, A, \Delta_4 \vdash D}{\Delta_1, C, \Delta_2, \Delta_3, \Delta_4 \vdash D}}{\Gamma, \Delta_1, \Delta_2, \Delta_3, \Delta_4 \vdash D}\text{cut} \quad \rightarrow$$

$$\dfrac{\dfrac{\Gamma \vdash C \qquad \Delta_1, C, \Delta_2 \vdash A}{\Gamma, \Delta_1, \Delta_2 \vdash A}\text{cut} \qquad \Delta_3, A, \Delta_4 \vdash D}{\Gamma, \Delta_1, \Delta_2, \Delta_3, \Delta_4 \vdash D}$$

The symmetrical case is similar.

Cut (case 2)

$$\dfrac{\dfrac{\Gamma \vdash A \qquad \Delta_1, A, \Delta_2 \vdash C}{\Gamma, \Delta_1, \Delta_2 \vdash C} \qquad \Delta_3, C, \Delta_4 \vdash D}{\Gamma, \Delta_1, \Delta_2, \Delta_3, \Delta_4 \vdash D}\text{cut} \quad \rightarrow$$

$$\dfrac{\Gamma \vdash A \qquad \dfrac{\Delta_1, A, \Delta_2 \vdash C \qquad \Delta_3, C, \Delta_4 \vdash D}{\Delta_1, A, \Delta_2, \Delta_3, \Delta_4 \vdash D}\text{cut}}{\Gamma, \Delta_1, \Delta_2, \Delta_3, \Delta_4 \vdash D}$$

Ex. (case 1)

$$\dfrac{\Gamma \vdash C \qquad \dfrac{\Delta_1, C, \Delta_2, A, B, \Delta \vdash D}{\Delta_1, C, \Delta_2, B, A, \Delta \vdash D}}{\Gamma, \Delta_1, \Delta_2, B, A, \Delta \vdash D}\text{cut} \quad \rightarrow \quad \dfrac{\dfrac{\Gamma \vdash C \qquad \Delta_1, C, \Delta_2, A, B, \Delta \vdash D}{\Gamma, \Delta_1, \Delta_2, A, B, \Delta \vdash D}\text{cut}}{\Gamma, \Delta_1, \Delta_2, B, A, \Delta \vdash D}$$

The symmetrical case is similar.

Ex. (case 2)

$$\dfrac{\Gamma \vdash A \qquad \dfrac{\Delta_1, A, B, \Delta_2 \vdash C}{\Delta_1, B, A, \Delta_2 \vdash C}}{\Gamma, \Delta_1, B, \Delta_2 \vdash C}\text{cut} \quad \rightarrow \quad \dfrac{\Gamma \vdash A \qquad \Delta_1, A, B, \Delta_2 \vdash C}{\Gamma, \Delta_1, B, \Delta_2 \vdash C}\text{cut}$$

Ex. (case 3)

$$\dfrac{\dfrac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C} \qquad \Delta_1, C, \Delta_2 \vdash D}{\Gamma, B, A, \Delta, \Delta_1, \Delta_2 \vdash D}\text{cut} \quad \rightarrow \quad \dfrac{\dfrac{\Gamma, A, B, \Delta \vdash C \qquad \Delta_1, C, \Delta_2 \vdash D}{\Gamma, A, B, \Delta, \Delta_1, \Delta_2 \vdash D}\text{cut}}{\Gamma, B, A, \Delta, \Delta_1, \Delta_2 \vdash D}$$

$\otimes$R

$$\dfrac{\Gamma \vdash C \qquad \dfrac{\Delta_1, C, \Delta_2 \vdash A \qquad \Delta_3 \vdash B}{\Delta_1, C, \Delta_2, \Delta_3 \vdash A \otimes B}}{\Gamma, \Delta_1, \Delta_2, \Delta_3 \vdash A \otimes B}\text{cut} \quad \rightarrow$$

$$\dfrac{\dfrac{\Gamma \vdash C \qquad \Delta_1, C, \Delta_2 \vdash A}{\Gamma, \Delta_1, \Delta_2 \vdash A}\text{cut} \qquad \Delta_3 \vdash B}{\Gamma, \Delta_1, \Delta_2, \Delta_3 \vdash A \otimes B}$$

The symmetrical case of $\otimes$R is similar and omitted.

$\otimes$L (case 1)

$$\dfrac{\Gamma \vdash C \qquad \dfrac{\Delta_1, C, \Delta_2, A, B \vdash D}{\Delta_1, C, \Delta_2, A \otimes B \vdash D}}{\Gamma, \Delta_1, \Delta_2, A \otimes B \vdash D}\text{cut} \quad \rightarrow \quad \dfrac{\dfrac{\Gamma \vdash C \qquad \Delta_1, C, \Delta_2, A, B \vdash D}{\Gamma, \Delta_1, \Delta_2, A, B \vdash D}\text{cut}}{\Gamma, \Delta_1, \Delta_2, A \otimes B \vdash D}$$

$\otimes$L (case 2)

$$\dfrac{\dfrac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C} \qquad \Delta_1, C, \Delta_2 \vdash D}{\Gamma, A \otimes B, \Delta_1, \Delta_2 \vdash D}\text{cut} \quad \rightarrow \quad \dfrac{\dfrac{\Gamma, A, B \vdash C \qquad \Delta_1, C, \Delta_2 \vdash D}{\Gamma, A, B, \Delta_1, \Delta_2 \vdash D}\text{cut}}{\Gamma, A \otimes B, \Delta_1, \Delta_2 \vdash D}$$

1L (case 1)

$$\dfrac{\Gamma \vdash C \qquad \dfrac{\Delta_1, C, \Delta_2 \vdash D}{\Delta_1, C, \Delta_2, \mathbf{1} \vdash D}}{\Gamma, \Delta_1, \Delta_2, \mathbf{1} \vdash D}\text{cut} \quad \rightarrow \quad \dfrac{\dfrac{\Gamma \vdash C \qquad \Delta_1, C, \Delta_2 \vdash D}{\Gamma, \Delta_1, \Delta_2 \vdash D}\text{cut}}{\Gamma, \Delta_1, \Delta_2, \mathbf{1} \vdash D}$$

1L (case 2)

$$\dfrac{\dfrac{\Gamma \vdash C}{\Gamma, \mathbf{1} \vdash C} \qquad \Delta_1, C, \Delta_2 \vdash D}{\Gamma, \mathbf{1}, \Delta_1, \Delta_2 \vdash D}\text{cut} \quad \rightarrow \quad \dfrac{\dfrac{\Gamma \vdash C \qquad \Delta_1, C, \Delta_2 \vdash D}{\Gamma, \Delta_1, \Delta_2 \vdash D}\text{cut}}{\Gamma, \mathbf{1}, \Delta_1, \Delta_2 \vdash D}$$

$\multimap$R

$$\dfrac{\Gamma \vdash C \qquad \dfrac{\Delta_1, C, \Delta_2, A \vdash B}{\Delta_1, C, \Delta_2 \vdash A \multimap B}}{\Gamma, \Delta_1, \Delta_2 \vdash A \multimap B}\text{cut} \quad \rightarrow \quad \dfrac{\dfrac{\Gamma \vdash C \qquad \Delta_1, C, \Delta_2, A \vdash B}{\Gamma, \Delta_1, \Delta_2, A \vdash B}\text{cut}}{\Gamma, \Delta_1, \Delta_2 \vdash A \multimap B}$$

$\multimap$L (case 1)

$$\dfrac{\Gamma \vdash C \qquad \dfrac{\Delta_1, C, \Delta_2 \vdash A \qquad \Delta_3, B \vdash D}{\Delta_1, C, \Delta_2, \Delta_3, A \multimap B \vdash D}}{\Gamma, \Delta_1, \Delta_2, \Delta_3, A \multimap B \vdash D}\text{cut} \quad \rightarrow$$

$$\dfrac{\dfrac{\Gamma \vdash C \qquad \Delta_1, C, \Delta_2 \vdash A}{\Gamma, \Delta_1, \Delta_2 \vdash A}\text{cut} \qquad \Delta_3, B \vdash D}{\Gamma, \Delta_1, \Delta_2, \Delta_3, A \multimap B \vdash D}$$

The symmetrical case of $\multimap$L is similar and omitted.

$\multimap$L (case 2)

$$\dfrac{\dfrac{\Gamma \vdash A \qquad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C} \qquad \Delta_1, C, \Delta_2 \vdash D}{\Gamma, \Delta, A \multimap B, \Delta_1, \Delta_2 \vdash D}\text{cut} \quad \rightarrow$$

$$\dfrac{\Gamma \vdash A \qquad \dfrac{\Delta, B \vdash C \qquad \Delta_1, C, \Delta_2 \vdash D}{\Delta, B, \Delta_1, \Delta_2 \vdash D}\text{cut}}{\Gamma, \Delta, A \multimap B, \Delta_1, \Delta_2 \vdash D}$$

!L (case 1)

$$\dfrac{\Gamma \vdash C \qquad \dfrac{\Delta_1, C, \Delta_2, A^{(n)} \vdash D}{\Delta_1, C, \Delta_2, !A \vdash D}\,!\mathrm{L}}{\Gamma, \Delta_1, \Delta_2, !A \vdash D}\,\mathrm{cut} \quad \rightarrow \quad \dfrac{\dfrac{\Gamma \vdash C \qquad \Delta_1, C, \Delta_2, A^{(n)} \vdash D}{\Gamma, \Delta_1, \Delta_2, A^{(n)} \vdash D}\,\mathrm{cut}}{\Gamma, \Delta_1, \Delta_2, !A \vdash D}\,!\mathrm{L}$$

!L (case 2)

$$\dfrac{\dfrac{\Gamma, A^{(n)} \vdash C}{\Gamma, !A \vdash C}\,!\mathrm{L} \qquad \Delta_1, C, \Delta_2 \vdash D}{\Gamma, !A, \Delta_1, \Delta_2 \vdash D}\,\mathrm{cut} \quad \rightarrow \quad \dfrac{\dfrac{\Gamma, A^{(n)} \vdash C \qquad \Delta_1, C, \Delta_2 \vdash D}{\Gamma, A^{(n)}, \Delta_1, \Delta_2 \vdash D}\,\mathrm{cut}}{\Gamma, !A, \Delta_1, \Delta_2 \vdash D}\,!\mathrm{L}$$

This exhausts all the cases for cuts on non-principal formulas.

Cut of principal formulas

In each of the following cases, the weight of the proof after the reduction is strictly less than the weight before the reduction.

Ax. versus any (case 1)

$$\dfrac{\dfrac{}{A \vdash A}\,\mathrm{Ax} \qquad \Gamma, A, \Delta \vdash C}{\Gamma, A, \Delta \vdash C}\,\mathrm{cut} \quad \rightarrow \quad \Gamma, A, \Delta \vdash C$$

Ax. versus any (case 2)

$$\dfrac{\Gamma \vdash A \qquad \dfrac{}{A \vdash A}\,\mathrm{Ax}}{\Gamma \vdash A}\,\mathrm{cut} \quad \rightarrow \quad \Gamma \vdash A$$

$\otimes$R versus $\otimes$L

$$\dfrac{\dfrac{\Gamma \vdash A \qquad \Delta_1 \vdash B}{\Gamma, \Delta_1 \vdash A \otimes B}\,\otimes\mathrm{R} \qquad \dfrac{\Delta_2, A, B \vdash C}{\Delta_2, A \otimes B \vdash C}\,\otimes\mathrm{L}}{\Gamma, \Delta_1, \Delta_2 \vdash C}\,\mathrm{cut} \quad \rightarrow$$

$$\dfrac{\Gamma \vdash A \qquad \dfrac{\Delta_1 \vdash B \qquad \Delta_2, A, B \vdash C}{\Delta_1, \Delta_2, A \vdash C}\,\mathrm{cut}}{\Gamma, \Delta_1, \Delta_2 \vdash C}\,\mathrm{cut}$$

1R versus 1L

$$\dfrac{\dfrac{}{\vdash 1} \quad \dfrac{\Gamma \vdash C}{\Gamma, 1 \vdash C}}{\Gamma \vdash C}\text{cut} \quad \rightarrow \quad \Gamma \vdash C$$

$\multimap$R versus $\multimap$L

$$\dfrac{\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \quad \dfrac{\Delta_1 \vdash A \quad \Delta_2, B \vdash C}{\Delta_1, \Delta_2, A \multimap B \vdash C}}{\Gamma, \Delta_1, \Delta_2 \vdash C}\text{cut} \quad \rightarrow$$

$$\dfrac{\dfrac{\Delta_1 \vdash A \quad \Gamma, A \vdash B}{\Gamma, \Delta_1 \vdash B}\text{cut} \quad \Delta_2, B \vdash C}{\Gamma, \Delta_1, \Delta_2 \vdash C}\text{cut}$$

!R versus !L

$$\dfrac{\dfrac{\Gamma \vdash A}{!\Gamma \vdash !A} \quad \dfrac{\Delta, A^{(n)} \vdash C}{\Delta, !A \vdash C}}{!\Gamma, \Delta \vdash C}\text{cut} \quad \rightarrow \quad \dfrac{\dfrac{\Gamma \vdash A \cdots \Gamma \vdash A \quad \Delta, A^{(n)} \vdash C}{\Gamma^{(n)}, \Delta \vdash C}\text{cut}}{!\Gamma, \Delta \vdash C}$$

!R versus !R

$$\dfrac{\dfrac{\Gamma \vdash A}{!\Gamma \vdash !A} \quad \dfrac{\Delta_1, A, \Delta_2 \vdash C}{!\Delta_1, !A, !\Delta_2 \vdash !C}}{!\Gamma, !\Delta_1, !\Delta_2 \vdash !C}\text{cut} \quad \rightarrow \quad \dfrac{\dfrac{\Gamma \vdash A \quad \Delta_1, A, \Delta_2 \vdash C}{\Gamma, \Delta_1, \Delta_2 \vdash C}\text{cut}}{!\Gamma, !\Delta_1, !\Delta_2 \vdash !C}$$

This exhausts all the cases for cuts on principal formulas. □

Lemma A.0.2. *If a sequent is provable in IMSLL with a proof of weight $d > 0$, then it is provable in IMSLL with a proof of weight less than d.*

Proof. By induction on the number of proof rules applied in the derivation.

Since the weight of the proof is greater than zero, there must be some cut in the proof. If the last rule is not cut, then by induction we may form proofs of its hypotheses of weight less than d. Applying the same rule to the resulting reduced weight hypotheses produces the desired proof of weight less than d.

On the other hand, if the last rule is cut, then by induction we may produce proofs of the hypotheses of the cut of weight less than d. By a single application of Lemma A.0.1 to the resulting proof constructed from the modified hypotheses, we obtain the desired proof of weight less than d. □

Theorem A.0.3. (Cut-elimination). *If a sequent is provable in IMSLL, then it is provable in IMSLL without using the cut rule.*

Proof. By induction on the weight of the assumed proof. We may apply Lemma A.0.2 at each inductive step, and at the base case the weight of the proof is zero, so therefore by definition of the proof weight there are no cuts, and we have our desired cut free proof. □

Appendix B

Table of Acronyms

The intuitionistic linear logic connectives are partitioned into the multiplicatives $(\otimes, \multimap)$, the additives $(\&, \oplus)$, and the exponential modality !. In each of the following cases, the prefix "I" may be added to denote the intuitionistic system and a subscript 2 may be added to denote the second order system.

Acronym	Meaning
LL	linear logic
AL	affine logic (= LL + full weakening)
SLL	soft linear logic
SAL	soft affine logic
LLL	light linear logic
LAL	light affine logic
ELL	elementary linear logic
EAL	elementary affine logic
BLL	bounded linear logic
MLL	multiplicative linear logic (no additives; no exponentials)
MAL	multiplicative affine logic (no additives; no exponentials)
MSLL	multiplicative soft linear logic (no additives)
MSAL	multiplicative soft affine logic (no additives)
MLAL	multiplicative light affine logic (no additives)
MELL	multiplicative exponential linear logic (no additives)

Bibliography

[1] S. Abramsky, *Semantics of Interaction: an introduction to Game Semantics*, In Proceedings of the 1996 CLiCS Summer School, Isaac Newton Institute, P. Dybjer and A. Pitts, eds., Cambridge University Press, pp 1–31, (1997).

[2] S. Abramsky, *Predicative Copying and Polynomial Time*, Clifford Lectures, Tulane, (2002). Slides available at:
http://web.comlab.ox.ac.uk/oucl/work/samson.abramsky/pcpt.pdf

[3] S. Abramsky, R. Jagadeesan, *Games and Full Completeness for Multiplicative Linear Logic*, Journal of Symbolic Logic 59(2), pp 543–574, (1994).

[4] S. Abramsky, R. Jagadeesan, P. Malacaria, *Full abstraction for PCF*, In Theoretical Aspects of Computer Software, Springer-Verlag, pp 1–15 (1994).

[5] A. Asperti, L. Roversi, *Intuitionistic light affine logic*, ACM Transactions on Computational Logic 1(3), pp 1–39, (2002).

[6] P. Baillot, *Stratified coherence spaces: a denotational semantics for Light Linear Logic*, Theoretical Computer Science 318(1-2), pp 29–55, (2004).

[7] S. Bellantoni, S. Cook, *A new recursion-theoretic characterization of the polytime functions*, Computational Complexity 2, pp 97–110, (1992).

[8] G.M. Bierman, *What is a Categorical Model of Intuitionistic Linear Logic?*, In Proceedings of Conference on Typed lambda calculus and Applications, Springer-Verlag, LNCS 902, pp 78–93, (1995).

[9] R. Blute, M. Hamano, P. Scott, *Softness of hypercoherences and MALL full completeness*, Annals of Pure and Applied Logic 131, pp 1–63, (2005).

[10] R. Blute, P. Scott, *Category Theory for Linear Logicians*, in Linear Logic in Computer Science, Cambridge University Press, pp 3–64, (2004).

[11] F. Borceux, *Handbook of Categorical Algebra 1 (Basic Category Theory)*, Cambridge University Press, (1994).

[12] S.R. Buss, *Bounded Arithmetic*, Ph.D. thesis, Princeton University, (1985).

[13] A. Cobham, *The intrinsic computational difficulty of functions*, In Interntional Congress for Logic, Methodology, and the Philosophy of Science, North-Holland, pp 24–30, (1964).

[14] J.R.B. Cockett, P. Hofstra, *Introduction to Turing categories*. Submitted.

[15] U. Dal Lago, M. Hofmann, *Quantitative Models and Implicit Complexity*, In Proceedings of Foundations of Software Technology and Theoretical Computer Science, Springer-Verlag, LNCS 3821, pp 189–200, (2005).

[16] U. Dal Lago, S. Martini, *Phase semantics and decidability of elementary affine logic*, Theoretical Computer Science 318(3), pp 409–433, (2004).

[17] J.-Y. Girard, *Linear logic*, Theoretical Computer Science 50, pp 1–102 (1987).

[18] J.-Y. Girard, *Light Linear Logic*, Information and Computation 143, pp 175–204 (1998).

[19] J.-Y. Girard, Y. Lafont, P. Taylor, *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science 7, Cambridge University Press, (1989).

[20] J.-Y. Girard, A. Scedrov, P. Scott, *Bounded linear logic: a modular approach to polynomial-time computability*, Theoretical Computer Science 97, pp 1–66, (1992).

[21] R. Goldblatt, *Topoi: the categorial analysis of logic*, Dover, (2006).

[22] J.R. Hindley, *BCK-Combinators and Linear λ-Terms have Types*, Theoretical Computer Science 64, pp 97–105, (1989).

[23] J.R. Hindley, J.P. Seldin, *Introduction to Combinators and λ-Calculus*, Cambridge University Press, (1986).

[24] M. Hofmann, *Type Systems for Polynomial-time Computation*, Habilitationsschrift, Darmstadt University of Technology, (1999).

[25] M. Hofmann, P. Scott, *Realizability Models for BLL-like Languages*, Theoretical Computer Science 318(1-2), pp 121–137, (2004).

[26] M. Kanovich, M.Okada, A. Scedrov, *Phase Semantics for Light Linear Logic*, Electronic Notes in Theoretical Computer Science 6, 14 pages (1997).

[27] Y. Lafont, *The finite model property for various fragments of linear logic*, Journal of Symbolic Logic 62, pp 1202–1208, (1997).

[28] Y. Lafont, *Soft linear logic and polynomial time*, Theoretical Computer Science 318, pp 163–180 (2004).

[29] J. Lambek, P.J. Scott, *Introduction to higher order categorical logic*, Cambridge University Press, (1986).

[30] O. Laurent, L. Tortora de Falco, *Obsessional cliques: a semantic characterization of bounded time complexity*, In Proceedings of LICS, IEEE Computer Society, pp 179–188, (2006).

[31] D. Leivant, J.-Y. Marion, *Lambda calculus characterizations of poly-time*, Fundamenta Informaticae 19(1-2), pp 167–184, (1993).

[32] P. Lincoln, J. Mitchell, A. Scedrov, N. Shankar, *Decision problems for propositional linear logic*, Annals of Pure and Applied Logic 56, pp 239–311, (1992).

[33] S. Mac Lane, *Categories for the Working Mathematician, 2nd Edition*, Springer, (1998).

[34] H. Mairson, K. Terui, *On the Computational Complexity of Cut-Elimination in Linear Logic*, In Proceedings of ICTCS, Springer-Verlag, LNCS 2841, pp 23–36, (2003).

[35] V. Mogbil, *Sémantique des phase, rèseaux de preuve et divers problèmes de décision en logique linéaire*, Ph.D. thesis, Université de la Méditerranée – Aix-Marseille II, (2001).

[36] A.S. Murawski, C.-H.L. Ong, *Discreet Games, Light Affine Logic and PTIME Computation*, In Proceedings of CSL, Springer-Verlag, LNCS 1862, pp 427–441, (2000).

[37] M. Okada, *Phase semantics for higher order completeness, cut-elimination and normalization proofs*, Electronic Notes in Theoretical Computer Science 3, (1996).

[38] C.H. Papadimitriou, *Computational Complexity*, Addison-Wesley, (1994).

[39] B.F. Redmond, *Multiplexor Categories and Models of Soft Linear Logic*, In Proceedings of Logical Foundations of Computer Science, Springer, LNCS 4514, pp 472–485, (2007).

[40] M. Sipser, *Introduction to the Theory of Computation*, PWS Publishing Company, (1997).

[41] R. Smullyan, *To Mock a Mockingbird*, Oxford University Press, (1990).

[42] K. Terui, *Proof Nets and Boolean Circuits*, in Proceedings of LICS, pp 182–191, (2004).