UNIVERSITÀ DEGLI STUDI DI TORINO

Université de Paris

UNIVERSITÀ ITALO FRANCESE

SCUOLA DI DOTTORATO	ÉCOLE DOCTORALE n^o 386
IN SCIENZE DELLA NATURA	SCIENCES MATHÉMATIQUES
E TECNOLOGIE INNOVATIVE	DE PARIS CENTRE
INDIRIZZO INFORMATICA	SPÉCIALITÉ INFORMATIQUE

Thèse de Doctorat en Cotutelle

Non-Laziness in Implicit Computational Complexity and Probabilistic λ-calculus

Gianluca Curzi

Directeur: Luca Roversi   Co-directeur: Michele Pagani

Présentée et soutenue publiquement le 12 juin 2020, devant un jury composé de:

M. Patrick Baillot (Examinateur)	Directeur de recherche, CNRS, ENS Lyon
M. Ugo Dal Lago (Rapporteur)	Professeur, Università Bologna
Mme Claudia Faggian (Examinatrice)	Chargée de recherche, CNRS, Université de Paris
M. Alessio Guglielmi (Rapporteur)	Professeur, University of Bath
M. Michele Pagani (Co-directeur)	Professeur, Université de Paris
M. Luca Roversi (Directeur)	Professeur, Università di Torino
M. Lorenzo Tortora De Falco (Examinateur)	Professeur, Università Roma Tre

---

Abstract

This thesis explores the benefits of non-laziness in both Implicit Computational Complexity and probabilistic computation. More specifically, this thesis can be divided in two main parts. In the first one, we investigate in all directions the mechanisms of linear erasure and duplication, which lead us to the type assignment systems LEM (*Linearly Exponential Multiplicative Type Assignment*) and LAM (*Linearly Additive Multiplicative Type Assignment*). The former is able to express weaker versions of the exponential rules of Linear Logic, while the latter has weaker additive rules, called *linear additives*. These systems enjoy, respectively, a cubic cut-elimination and a linear normalization result. Since linear additives do not require a lazy evaluation to avoid the exponential blow up in normalization (unlike the standard additives), they can be employed to obtain an implicit characterization of the functions computable in probabilistic polynomial time that does not depend on the choice of the reduction strategy. This result is achieved in $\mathsf{STA}_{\oplus}$, a system that extends STA (*Soft Type Assignment*) with a randomized formulation of linear additives. Also, this system is able to capture the complexity classes PP and BPP. The second part of the thesis is focused on the probabilistic λ-calculus endowed with an operational semantics based on the head reduction, i.e. a non-lazy call-by-name evaluation policy. We prove that probabilistic applicative bisimilarity is fully abstract with respect to context equivalence. This result witnesses the discriminating power of non-laziness, which allows to recover a perfect match between the two equivalences that was missing in the lazy setting. Moreover, we show that probabilistic applicative similarity is sound but not complete for the context preorder.

Résumé de thèse

Cette thèse explore les avantages de la “non-paresse” dans la Complexité Computationnelle Implicite et dans le lambda calcul probabiliste. Plus précisément, cette thèse peut être divisée en deux parties principales. Dans la première, nous étudions dans tous les sens les mécanismes d'effacement et de duplication linéaire, à la fois dans le lambda calcul linéaire et dans le système IMLL_2. Cette analyse nous conduit au système LEM (*Linearly Exponential Multiplicative Type Assignment*), capable d'exprimer des versions plus faibles des règles exponentielles de la Logique Linéaire. LEM satisfait la propriété de réduction du sujet et une forme “paresseux” d'élimination des coupures qui prend du temps cubique. Nous définissons également une traduction de LEM à IMLL_2. Cette traduction nous montre que les règles exponentielles de LEM compressent exponentiellement les mécanismes d'effacement et de duplication linéaire de IMLL_2. Enfin, nous explorons les avantages de cette compression, en codant à la fois les circuits booléens et les nombres naturels de manière compacte. De plus, nous introduisons un deuxième système, LAM (*Linearly Additive Multiplicative Type Assignment*), équipé d'une version plus faible des règles additives de la Logique Linéaire, appelées *additifs linéaires.* Ce système satisfait une normalisation linéaire, évitant ainsi le problème de l'explosion exponentielle des additifs standard sans utiliser de stratégies de réduction paresseuse. Nous étudions également une traduction de LAM en IMLL_2 similaire à celle de LEM. Étant donné que les additifs linéaires sont “inoffensifs” du point de vue de la complexité computationnelle, ils seront utilisés pour obtenir une caractérisation implicite des fonctions calculables en temps polynomial probabiliste qui ne dépend pas du choix des stratégies de réduction. Ce résultat a été réalisé dans $\mathsf{STA}_\oplus$, un système obtenu en dotant STA (*Soft Type Assignment*) d'une formulation probabiliste des additifs linéaires. $\mathsf{STA}_\oplus$ est un système de typage pour un calcul des termes confluent, et satisfait la propriété de réduction du sujet. Enfin, nous montrerons que $\mathsf{STA}_\oplus$ capture les classes de complexité probabiliste PP (*Probabilistic Polynomial time*) et BPP (*Bounded-error Probabilistic Polynomial time*), même si la caractérisation du BPP est moins “implicite”. La deuxième partie de la thèse se concentre sur le lambda calcul probabiliste non typé $\Lambda_\oplus$ doté d'une sémantique opérationnelle basée sur la réduction de tête, c'est-à-dire une politique d'évaluation non-paresseuse de l'appel par nom. Nous prouverons que la bisimilarité probabiliste est “fully abstract” par rapport à l'équivalence observationnelle. Ce résultat témoigne le pouvoir discriminant de la non-paresse, qui permet de retrouver une correspondance parfaite entre les deux équivalences, qui manquait dans le cadre paresseuse. Plus précisément, la sémantique opérationnelle que nous donnerons implémentera la soi-disant “head spine reduction”, une variante de la réduction de tête. Nous montrerons donc que les deux stratégies sont équivalentes, c'est-à-dire que la probabilité qu'un terme converge vers une forme de tête normale donnée est la même pour les deux stratégies de réduction. D'une côté, la head spine reduction nous permettra de prouver plus facilement le théorème de correction. D'autre côté, l'équivalence entre les deux stratégies de réduction nous permettra de montrer le théorème de complétude en utilisant le théorème de séparation de Leventis dans le contexte des arbres de Nakajima probabilistes. Enfin, nous montrerons que la similarité probabiliste est correcte mais pas complète par rapport à le préordre observationnel à travers un contre-exemple, en discutant également comment un résultat de complétude pourrait être obtenu dans un calcul étendu avec le “parallel or” de Plotkin.

Acknowledgements

I would like to sincerely express my gratitude to my supervisor Luca Roversi and my co-supervisor Michele Pagani for their precious and constant help and for having transmitted to me their enthusiasm in doing research.

I am also grateful to Ugo Dal Lago and Alessio Guglielmi, who kindly accepted to review this thesis.

I owe a great debt to Felice Cardone and Luca Paolini, for their stimulating discussions about category theory and the foundations of computing, that have been source of great inspiration to me.

I would like to thank Francesco, Stefano, Noemi, Valentina, Milena, Antonio, Livio, Elena, Ruggero (and whoever I forgot to report). I spent valuable moments with all of them. I am also grateful to all my colleagues in Paris. I am particularly indebted with Jules, Leonard and Daniel, who passionately helped me in learning French (my poor French!). I had intriguing conversations with them about science, logic and mathematics that made unforgettable my stay in Paris.

A personal thank goes to my family, who always and unconditionally supports me, and to my friends Gianluca, Matteo and Gabriele. Finally, a special acknowledgement goes to Federica, whose infinite patience and continuous encouragement throughout these years made this thesis possible.

Contents

Chapter 1

Introduction

This thesis concerns two distinct topics in theoretical computer science that quite recently started interacting with fruitful implications, namely *Implicit Computational Complexity* (ICC) and *probabilistic computation.* ICC is a branch of computational complexity originated in the nineties and its goal is to capture the inherent principles of bounded computation by means of languages or calculi that do not directly rely on machine models nor explicit restrictions on resources. A special attention in ICC is paid to those computational problems which are "tractable", i.e. that can be solved efficiently, requiring for example a polynomial amount of time with respect to the input size. Indeed, one of the purposes of ICC is to suggest new techniques to statically verify the runtime of a program, where efficiency is a *desideratum.*

The need for faster algorithms to solve real world problems has attracted along the years a growing interest in probabilistic programming, where random choices are permitted during execution. Randomized programs can be more efficient with respect to deterministic programs, at the cost of allowing wrong answers. One of the main issues in the design of randomized algorithms is to obtain a highly trustworthy output.

The degree of accuracy in returning the correct answer is among the features discriminating a probabilistic complexity class. For example, the class BPP (Bounded-error Probabilistic Polynomial time) collects all problems solvable by randomized algorithms running in polynomial time with error probability bounded by a constant strictly smaller than a half. A striking aspect of this class is that the error probability can in principle be reduced at will while incurring only a polynomial slowdown, so increasing the reliability of the answer without affecting the efficiency.

Starting from Mitchell et al. [72], several attempts have been made to capture the probabilistic polynomial time in the style of ICC by means of higher-order languages. Examples are Zhang [95] or Dal Lago and Toldin [28]. In particular, the latter work also discusses the inherent difficulties of characterizing the class BPP implicitly, due to the presence of external error bounds. Recently, Seiller has proposed a promising semantic approach to ICC based on the notion of Interaction Graphs [83], showing how to capture the classes PL (Probabilistic Logarithmic space) and PP (Probabilistic Polynomial time).

Beside probabilistic complexity, the pervasive role of stochastic models in computer science, like in natural language processing [66], machine learning [75], formal verification [31] and cryptography [47], has also stimulated foundational research in probabilistic λ-calculi, with a special focus on program equivalence. Different operational techniques have been employed for understanding and reasoning about program equivalence, like *context equivalence*, which identifies programs "behaving" the same in any possible programming context, or *bisimilarity*, identifying programs that can "simulate" each other. The latter notion was previously studied in concurrency theory to equate processes and exploits coinductive proof methods, which are subject of growing attention in the literature [80, 81].

A common thread in program equivalence and ICC is the crucial role played by the reduction strategies, especially in a (probabilistic) higher-order setting. On the one hand, recovering confluence in probabilistic higher-order calculi requires a specific calling mechanism to discriminate between duplicating a function which performs a choice and duplicating the choice [29]. On the other hand, both the notions of program equivalence and of implicit characterization are sensitive to the choice of the evaluation policy. Concerning the former, it has been shown for example that bisimilarity is strictly finer than context equivalence in the (lazy) call-by-name probabilistic λ-calculus [27], while the two relations perfectly match in the call-by-value setting [22]. Concerning implicit complexity, both PSPACE (Polynomial Space) and NPTIME (Non-deterministic Polynimial time) have been characterized by means of extensions of the type assignment system STA (Soft Type Assignment) [38], but special reduction strategies are required to avoid an exponential blow up in normalization. These complexity classes cannot be captured by an innermost evaluation policy, because this would let the size of a term and the number of its redexes grow exponentially. A variant of the leftmost outermost strategy is needed, which delays the substitutions coming from β-reduction as long as possible.

A discriminating aspect in the choice of the reduction strategy in (probabilistic) higher-order calculi is *laziness*. It allows to "freeze" the evaluation, typically inside a constructor, according to the principle of producing as little information as possible at each step of the computation. In ICC, lazy evaluation has been employed to recover a polytime normalization in presence of additive-like rules [44]. In the (probabilistic) λ-calculus lazy reduction strategies, such as (lazy) call-by-name or call-by-value, forbid evaluation inside the scope of any abstraction, so that $\lambda x.M$ is a value whatever M is. By contrast, in a non-lazy policy, such as non-lazy call-by-name (often called head reduction), we keep reducing what is inside the abstraction. Switching from a lazy to a non-lazy strategy we affect the notion of program equivalence, whether this is defined in terms of context equivalence or bisimilarity.

The fundamental objective of this thesis is to show the benefits of being "non-lazy" both in ICC and in the probabilistic λ-calculus:

- on the one hand, the choice of a lazy strategy to prevent exponential explosions in presence of the additive rules can be avoided by adopting a weaker notion of additive relying on Mairson and Terui's linear erasure and duplication [64, 65], that can be used for implicit characterizations of the probabilistic polynomial time regardless of the reduction strategy considered;

- on the other hand, "non-laziness" is the key step to recover the missing match between bisimilarity and context equivalence in the call-by-name probabilistic λ-calculus.

1.1 Contributions

In this section we present the main contributions of each chapter.

Chapter 3.

- We give a complete and detailed proof of the Duplication Theorem (Section 3.2), that assures the existence of a linear "duplicator" for all closed and normal inhabitants of a special class of types in IMLL_2, representing finite data types. This proof was only sketched by Mairson and Terui in [65], and amounts to show the existence of two linear λ-terms in

IMLL_2: a "compiler", able to map each closed and normal inhabitant M into a linear λ-term $\lceil M \rceil$ representing the encoding of M, and a "decoder", able to map $\lceil M \rceil$ back to a pair containing two copies of M. Constructing the duplicator is a hard task, essentially because the language of IMLL_2 is quite poor, and requires several preliminary results. In particular, we show in Lemma 36 of Section 3.3.4 that the size of a duplicator of type A is exponential in the size of A. We widely apply the Duplication Theorem in both Chapter 3 and Chapter 4, because it allows to express a weaker form of contraction.

- We introduce a new system, LEM (*Linearly Exponential Multiplicative Type Assignment*), that extends IMLL_2 with primitive rules for Mairson and Terui's linear weakening and contraction [65], so compressing in a single step the exponential mechanism of duplication in IMLL_2 (see the previous point). A fundamental advantage of this compression is that LEM allows for more compact and modular representations of boolean circuits and natural numbers, as opposed to their respective encodings in IMLL_2 [65, 64].

Chapter 4.

- We introduce a new system, LAM (*Linearly Additive Multiplicative Type Assignment*), that extends IMLL_2 with a weaker version of the additive rules called *linear additives*, which are based on the mechanisms of linear weakening and contraction [65]. LAM can be seen as a restricted formulation of IMALL_2. In the latter, normalization can be exponential (both in time and space), and one can recover a linear normalization only by adopting specific reduction strategies, like *lazy reduction* (see [45, 44]). By contrast LAM enjoys a linear *strong* normalization, so that lazy reduction is not needed.
- We present a new system, $\mathsf{STA}_{\oplus}$, that combines a probabilistic version of linear additives (see the previous point) with STA (Soft Type Assignment) [40], and is able to capture the probabilistic polynomial time functions as well as the classes PP and BPP: on the one hand, linear additives are costless from a complexity-theoretic viewpoint, so the system inherits in a natural way the polynomial bound on normalization from STA; on the other hand, they are expressive enough to encode the transition function of a Probabilistic Turing Machine, a fundamental ingredient to obtain polytime completeness. To our knowledge, this is the first characterization result in a probabilistic setting that is close to the style of light logics. Previous characterizations have been achieved through extensions of SLR (Safe Linear Recursion) [49], as for example [95, 28]. Moreover, as opposed to such previous works, our characterization does not depend on the choice of a reduction strategy.

Chapter 5.

- We prove that the head reduction and the head *spine* reduction [84] are equivalent in the probabilistic λ-calculus, meaning that the probability that a given term M converges to a certain head normal form in exactly n steps of reduction is the same for both evaluation strategies (Section 5.2). As a straightforward consequence, we get a similar equivalence in the standard λ-calculus: a term M converges to a head normal form H in exactly n steps according to head reduction if and only if it does so according to head spine reduction. As far as we know, this result is not in the literature, even in the deterministic case. Switching from head reduction to head spine reduction can be often convenient. For example, the big-step presentation of the head spine reduction is more compact with respect to head reduction, the latter requiring a further big-step relation based on (lazy) call-by-name evaluation (relating terms with distributions of weak head normal forms), so doubling the number of rules (see Section 5.1.2).

- We prove that probabilistic applicative bisimilarity is fully abstract with respect to non-lazy call-by-name (or head) context equivalence in the probabilistic λ-calculus (Theorem 134 of Section 5.4.2). In a sense, this theorem completes the picture by providing the missing coinductive characterization to the previously established full abstraction results concerning denotational models [20, 61], game semantics [20], and probabilistic Nakajima trees [60, 61]. In particular, as a straightforward consequence of Theorem 134 and [92], we can infer that testing equivalence is fully abstract for the head context equivalence. Also, we show that probabilistic applicative similarity is sound but not complete (hence fully abstract) with respect to the head context preorder, the latter result being achieved by means of a counterexample.

1.2 Content of the thesis

This thesis consists of two main parts. The first one is focused on ICC and investigates several type assignment systems based on Linear Logic, while the second one concerns the probabilistic λ-calculus and the related operational techniques for program equivalence. In the following, we offer a brief overview of the content of each chapter.

Chapter 3. We study erasability and duplication in the linear λ-calculus and in the corresponding type assignment IMLL_2. Then, we introduce the type system LEM, able to exponentially compress Mairson and Terui's mechanisms of linear weakening and contraction [65], and we establish some basic computational and proof-theoretical properties, like subject reduction and a "lazy" form of cut-elimination. Also, we define a translation of the system into IMLL_2, proving a simulation property. Last, we explore some applications of LEM by encoding in a compact way boolean circuits and natural numbers.

Chapter 4. In order to deal with additive rules without incurring an exponential blow up in normalization, we designed LAM, a type system based on Mairson and Terui's linear weakening and contraction [65]. LAM is endowed with restricted additive rules called *linear additives*, that enjoy a linear time normalization. Then we present a probabilistic version of STA [40], called $\mathsf{STA}_\oplus$, whose probabilistic features are given by a non-deterministic variant of linear additives. $\mathsf{STA}_\oplus$ is able to capture in a natural way the probabilistic polynomial time as well as the classes PP and BPP.

Chapter 5. We present the untyped probabilistic λ-calculus $\Lambda_\oplus$ endowed with an operational semantics based on the head *spine* reduction that, as we show, is an equivalent variant of the head reduction. Then prove that probabilistic applicative bisimilarity is fully abstract with respect to context equivalence. Soundness is established by means of a context lemma, while completeness relies on a fundamental separation result from [60]. Last, we show that probabilistic applicative similarity is sound but not complete with respect to context preorder using a counterexample.

Chapter 2

Background

In the following sections we give an overview of the basic notions and results from the main areas of computer science this thesis is based on, such as the Curry-Howard isomorphisms paradigm, Linear Logic, Implicit Computational Complexity, bisimulation and coinductive methods. We conclude by introducing some preliminary definitions and notational conventions.

2.1 The Curry-Howard isomorphism

2.1.1 Church's λ-calculus

The *λ-calculus* is a model of computation introduced by Alonzo Church in the 1930s. It is defined as a formal language of *λ-terms*, endowed with a rewriting rule called *β-reduction*. The λ-terms are generated by the following grammar:

$$M := \ x \mid \lambda x.M \mid MM$$

where x is taken from a denumerable set of *variables*. The expression $\lambda x.M$ is called *abstraction* and binds the variable x, while the expression MN is called *application*. The β-reduction rule is the following:

$$(\lambda x.M)N \to_\beta M[N/x]$$

where $M[N/x]$ denotes the meta-level substitution of N for the free occurrences of the variable x in M.

Roughly, an abstraction $\lambda x.M$ represents a function, the expression MN represents the application of the function M to an argument N, and β-reduction "computes" the result of the function application $(\lambda x.M)N$. To motivate this intuition, we consider a simple example.

Example 1. Suppose we are given the polynomial $x^2 - 2x + 5$ and we want to evaluate it when $x = 2$. In this case we replace x with 2 in the expression and we obtain $2^2 - 2 \cdot 2 + 5$. By carrying out some basic arithmetic operations, we eventually get the value 5. In the λ-calculus, this procedure can be done in three steps. First, we use the λ-operator to turn the expression $x^2 - 2x + 5$ into a function $M \triangleq \lambda x.(x^2 - 2x + 5)$, which *abstracts* over x and waits for a value replacing the variable (in our case an integer). Then we *apply* the function M to the *argument* 2, i.e. we consider the term $M2$. Last, we compute $M2$ by applying the β-reduction:

$$(\lambda x.(x^2 - 2x + 5))\,2 \to_\beta 2^2 - 2 \cdot 2 + 5$$

that gives 5 as a result.

The above example mixes λ-terms with "external" mathematical functions, such as additions and products. However, despite its simplicity, the λ-calculus is able to express all arithmetic operations, so that $M2$ in Example 1 can be turned into a λ-term, and the evaluation of $M2$ can be entirely represented as a sequence of β-reduction steps. Actually, the λ-calculus is *Turing-complete*, i.e. powerful enough to encode all number-theoretic functions that are computable by a Turing Machine.

2.1.2 Simply typed λ-calculus

In mathematics, the definition of a particular function usually includes a statement of the kind of inputs it will accept, and the kind of outputs it will produce. For example, the squaring function accepts integers n as inputs and produces integers n^2 as outputs, and the zero-test function accepts integers and produces Boolean values ("true" or "false" according as the input is zero or not).

Corresponding to this way of defining functions, the λ-calculus can be modified by attaching *types* to λ-terms, i.e. labels denoting their input and output sets. The most elementary examples of types are called *simple types*, and are generated by the following grammar:

$$\sigma := \ \alpha \mid A \to A$$

where α is taken from a denumerable set of *type variables*, and $A \to B$ is called *function type.*

There are two different strategies in attaching types to terms, that give rise to different systems: the *Church-style simply typed λ-calculus*, or $\lambda_{\to}^{Ch}$ for short, and the *Curry-style simply typed λ-calculus*, or $\lambda_{\to}^{Cu}$ for short. In $\lambda_{\to}^{Ch}$ types are introduced in the very definition of λ-term:

- each λ-variable x is associated with a type A, written x^A;
- given x^A and M^B we construct the typed λ-term $(\lambda x^A.M^B)^{A\to B}$;
- given $M^{A\to B}$ and N^A, we construct the typed λ-term $(M^{A\to B}N^A)^B$.

In $\lambda_{\to}^{Ch}$ each variable comes with a type, so that the type of a λ-term is fixed. For example, $(\lambda x^A.x^A)^{A\to A}$ is the identity function with type $A \to A$. A different approach is adopted in the Curry-style simply typed λ-calculus, where $\lambda x.x$ is a general identity function that can receive any type of shape $C \to C$. In this case, a type A is *assigned* to a λ-term M, written $M : A$, with the purpose of giving informations about the shape of terms it can "safely" be applied to.

The most basic assignment in $\lambda_{\to}^{Cu}$ is called *declaration.* A declaration is an expression $x : A$, where x is a λ-variable, called *subject*, and A is a type, called *predicate.* Starting with declarations we can construct derivation trees assigning types to terms by applying the following rules:

$$\dfrac{\begin{array}{c}[x:A]\\ \vdots\\ M:B\end{array}}{\lambda x.M : A \to B}\ abs \qquad\qquad \dfrac{M : A \to B \qquad N : A}{MN : B}\ app$$

where the square brackets are added to *all* declarations of the shape $x : A$ (if any) above the place where the instance of *abs* is applied, and identify that all such $x : A$ are "discharged", i.e. no longer available.

Example 2. The following is a derivation tree of $\lambda xy.x : A \to B \to A$:

$$\cfrac{\cfrac{[x : A]}{\lambda y.x : B \to A}\,abs}{\lambda xy.x : A \to B \to A}\,abs$$

notice that the declaration $y : B$ has been "vacuously" discharged.

In the Curry-style simply-typed λ-calculus there exist λ-terms that receive no type, like the "self-application" $\lambda x.xx$, and the same happens in $\lambda^{Ch}_{\to}$.

2.1.3 Gentzen's proof systems

In 1935 Gerhard Gentzen [41] introduced two new proof systems for logic, called *natural deduction* and *sequent calculus*, which are now considered as standard in proof theory [90]. The starting point of his work was the analysis of mathematical proofs as they occur in practice. Gentzen pointed out that Hilbert's axiomatic formulation of logic cannot suitably express mathematical reasoning, and proposed natural deduction as a rigorous (but natural) counterpart for it.

We now briefly recall the natural deduction presentation of the implicative fragment of intuitionistic logic (system $\mathsf{NJ}_{\Rightarrow}$). Its formulas are generated by the following grammar:

$$A := \ \alpha \mid A \Rightarrow A$$

where α is taken from a denumerable set of propositional variables. In natural deduction we don't have axioms, but only assumptions and inference rules. An assumption is an expression A^x, were A is a formula, and x is taken from a denumerable set of variables. Starting from the assumptions we can construct derivation trees by applying instances of the following inference rules:

$$\cfrac{\begin{matrix}[A^x]\\ \vdots \\ B\end{matrix}}{A \Rightarrow B}\Rightarrow\mathrm{I} \qquad\qquad \cfrac{A \Rightarrow B \qquad A}{B}\Rightarrow\mathrm{E}$$

where the square brackets are added to *all* assumptions of the shape A^x (if any) above the place where the instance of $\Rightarrow$I is applied, and identify that all A^x are "discharged", i.e. no longer available.

In natural deduction, each connective comes with two rules, one introducing it and the other eliminating it. In $\mathsf{NJ}_{\Rightarrow}$ the introduction rule for the implication is $\Rightarrow$I, its elimination rule is $\Rightarrow$E.

Example 3. The following is a derivation of $(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow (A \Rightarrow C)$:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{(A \Rightarrow B \Rightarrow C)^x \qquad A^z}{B \Rightarrow C}\Rightarrow\mathrm{E} \qquad \cfrac{(A \Rightarrow B)^y \qquad A^z}{B}\Rightarrow\mathrm{E}}{C}\Rightarrow\mathrm{E}}{A \Rightarrow C}\Rightarrow\mathrm{I}(z)}{(A \Rightarrow B) \Rightarrow A \Rightarrow C}\Rightarrow\mathrm{I}(y)}{(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C}\Rightarrow\mathrm{I}(x)$$

where the notation (x) is used to highlight the assumption packet that has been discharged.

Gentzen noticed that natural deduction derivations may contain "detours", i.e. redundant steps where a connective is first introduced and then eliminated. To remove all detours from a derivation, he defined a procedure of "detour conversion". The typical detour for the implication, and its related conversion, are the following:

$$\dfrac{\dfrac{\begin{matrix}[A^x]\\ \mathcal{D}\\ B\end{matrix}}{A \Rightarrow B}\Rightarrow\mathrm{I} \qquad \begin{matrix}\mathcal{D}'\\ A\end{matrix}}{B}\Rightarrow\mathrm{E} \qquad \rightsquigarrow \qquad \begin{matrix}\mathcal{D}'\\ A\\ \mathcal{D}\\ B\end{matrix}$$

in which each occurrence of the assumption $[A^x]$ in the derivation $\mathcal{D}$ of B is replaced by a separate copy of the derivation $\mathcal{D}'$ of A. The process of eliminating detours is called *proof normalization*, and a proof tree with no detour is called *normal form*.

Gentzen has shown that every natural deduction derivation of intuitionistic logic can be turned into a normal form by normalization. Unfortunately, he was unable to extend this result to classical logic. This led him to devise a second formalism: sequent calculus. This proof system does not derive formulas, but *sequents*, i.e. expressions of the form $\Gamma \vdash C$, where C is a formula and Γ is a multiset of formulas. Intuitively, a sequent $\Gamma \vdash C$ represents a derivation of C from the assumptions in Γ.

The sequent calculus presentation of the implicative fragment of intuitionistic logic (system $\mathsf{LJ}_{\Rightarrow}$) is defined by the following rules:

$$\dfrac{}{x : A \vdash x : A}\,ax \qquad \dfrac{\Gamma \vdash A \qquad \Delta, A \vdash C}{\Gamma, \Delta \vdash C}\,cut \qquad \dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\Rightarrow\mathrm{R} \qquad \dfrac{\Gamma \vdash A \qquad \Delta, B \vdash C}{\Gamma, A \Rightarrow B, \Delta \vdash C}\Rightarrow\mathrm{L}$$

The rule *ax* represents a natural deduction assumption, while the rules ⇒R and ⇒L are the sequent calculus counterparts of the rules ⇒I and ⇒E, respectively. Last, the rule *cut* plays the role of detours, because it "cuts" the formula A appearing in both premises.

The fundamental result of Gentzen is the *cut-elimination theorem*, the analogous of proof normalization in natural deduction, assuring that any sequent calculus derivation can be turned into a cut-free one. As an immediate corollary of this theorem, Gentzen showed the existence of unprovable formulas in the system, yielding a new technique for establishing logical consistency.

2.1.4 Between logic and computation

A fascinating aspect in theoretical computer science is the correspondence between various typed λ-calculi and systems of formal logic, widely known as the *Curry-Howard isomorphism.*

This analogy was first noticed as a curious fact by Curry in 1934 [25, 26]. According to his intuition, the function type "$\rightarrow$" can be seen as an implication "$\Rightarrow$" in $\mathsf{NJ}_{\Rightarrow}$. On the one hand, the rule ⇒I "constructs" a function that maps any element of A into an element of B. On the other hand, the rule ⇒I "applies" a function $f : A \rightarrow B$ to an element of A, yielding an element of B.

Curry's intuition was later refined by William Howard in 1969 (his ideas were reported in a manuscript published in 1980 [51]). Howard understood that the connection between logic and computation is a fundamental principle or a paradigm, rather than a mere curiosity. He showed that the simply typed λ-calculus and the implicative fragment of intuitionistic logic perfectly match:

- a formula can be seen as a type, where the intuitionistic implication "$\Rightarrow$" is syntactic sugar for the function type "$\rightarrow$";
- a proof of A can be seen as a λ-terms with type A, where an assumption A^x is syntactic sugar for the declaration $x : A$, the rule ⇒I corresponds to an abstraction, and the rule ⇒E corresponds to an application;
- finally, proof normalization corresponds to β-reduction.

The example below discuss the isomorphisms between $\mathsf{NJ}_{\Rightarrow}$ and $\lambda_{\rightarrow}^{Cu}$ (a similar connection can be established for $\lambda_{\rightarrow}^{Ch}$ as well).

Example 4. Let us consider a derivation of $\mathsf{NJ}_{\Rightarrow}$:

$$\cfrac{\cfrac{\cfrac{\cfrac{[(A \Rightarrow B \Rightarrow A)^z] \qquad [A^{x'}]}{B \Rightarrow A}\Rightarrow\mathrm{E} \qquad [B^{y'}]}{A}\Rightarrow\mathrm{E}}{(A\Rightarrow B\Rightarrow A)\Rightarrow A}\Rightarrow\mathrm{I}(z) \qquad \cfrac{\cfrac{[A^x]}{B\Rightarrow A}\Rightarrow\mathrm{I}(y)}{A\Rightarrow B\Rightarrow A}\Rightarrow\mathrm{I}(x)}{A}\Rightarrow\mathrm{E}$$

This derivation corresponds to the λ-term $(\lambda z.zx'y')(\lambda xy.x)$ with type A in $\lambda_{\rightarrow}^{Cu}$. Now, if we apply the proof normalization, we obtain:

$$\cfrac{\cfrac{\cfrac{\cfrac{[A^x]}{B\Rightarrow A}\Rightarrow\mathrm{I}(y)}{A\Rightarrow B\Rightarrow A}\Rightarrow\mathrm{I}(x) \qquad [A^{x'}]}{B\Rightarrow A}\Rightarrow\mathrm{E} \qquad [B^{y'}]}{A}\Rightarrow\mathrm{E}$$

which corresponds to an application of the β-reduction step $(\lambda z.x'y')(\lambda xy.x) \rightarrow_{\beta} (\lambda xy.x)x'y'$. The above derivation eventually reduces by proof normalization to a single assumption of $A^{x'}$, which corresponds to the β-reduction steps $(\lambda xy.x)x'y' \rightarrow_{\beta} (\lambda y.x')y' \rightarrow_{\beta} x'$ leading to a variable x' of type A, i.e. a declaration $x' : A$.

Since the 1969 work, various styles of presentations of logical systems (Hilbert style, natural deduction, sequent calculus) have been shown to correspond to different models of computations (combinatory logic, λ-calculus, explicit substitution calculi). Moreover, the Curry-Howard isomorphism has been gradually extended to give a computational interpretation of several proof-theoretical concepts. For example, quantification in predicate logic corresponds to dependent product, second-order logic is connected to polymorphism, and proofs by contradiction in classical logic are connected to control operators).

2.2 Linear Logic

2.2.1 Toward Linear Logic

Linear logic has been introduced by Jean-Yves Girard in his seminal work [46] as a refinement of both classical and intuitionistic logic, able to combine the dualities of the former with the constructive nature of the latter.

To motivate the need for this new logic, we start by analysing the structural rules *weakening* and *contraction* in intuitionistic logic. These can be displayed in sequent calculus style as follows:

$$\frac{\Gamma \vdash B}{\Gamma, A \vdash B}\,\mathit{weak} \qquad\qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\,\mathit{contr}$$

As Girard mentioned in [43], the weakening rule "opens the door to fake dependencies", because it breaks the relevance relations between the assumptions and the conclusions, while the contraction is the "fingernail of infinity", because it allows the multiple use of an assumption. In a word, these rules are bad bookkeepers, since they make us loose control on our expenses, and hence waste our resources.

The intuitionistic presentation of Linear Logic is the outcome of two fundamental operations:

- we forbid the unrestricted applications of weakening and contraction;
- we recover in a controlled way the full expressiveness of intuitionistic logic by adding four logical rules, p (*promotion*), d (*dereliction*), w (*weakening*) and c (*contraction*), that keep track of the reuse of resources by means of a new modality "!" (*of course* or *exponential modality*):

  $$\frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\,p \qquad \frac{\Gamma, A \vdash C}{\Gamma, !A \vdash C}\,d \qquad \frac{\Gamma \vdash C}{\Gamma, !A \vdash C}\,w \qquad \frac{\Gamma, !A, !A \vdash C}{\Gamma, !A \vdash C}\,c$$

  these are also called *exponential rules.*

Therefore, the modality ! allows to carefully handle resources introducing a neat distinction between those assumptions which are *linear*, i.e. consumable exactly once, and those which are *reusable* by means of the exponential rules.

The lack of unrestricted weakening and contraction has several consequences. First, Linear Logic is able to decompose the usual intuitionistic implication $A \Rightarrow B$ into $!A \multimap B$, where $\multimap$ is the *linear implication* and the modality ! allows the multiple or vacuous use of the assumption A to get the conclusion B.

Another fundamental consequence is the presence of two distinct versions of the conjunction $\wedge$, i.e. $\otimes$ (*multiplicative conjunction* or *tensor*) and & (*additive conjunction* or *with*), two distinct versions for the disjunction $\vee$, i.e $\invamp$ (*multiplicative disjunction* or *par*) and $\oplus$ (*additive disjunction* or *plus*), and four distinct units:

	multiplicative	additive
$\wedge$	$\otimes$	&
$\vee$	$\invamp$	$\oplus$
true	$\mathbf{1}$	$\top$
false	$\bot$	$\mathbf{0}$

Let us explain why this happens. In intuitionistic logic both $\wedge$ and $\vee$ have a *multiplicative* and an *additive* formulation. As an example, the multiplicative version of $\wedge$ has a right rule $\wedge$R with two premises, each one with a different context, and a left rule $\wedge$L with a single premise:

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B}\,\wedge\mathrm{R} \qquad\qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}\,\wedge\mathrm{L}$$

while the additive version of $\wedge$ has a right rule $\wedge$R with two premises sharing the same context, and two left rules $\wedge$L1 and $\wedge$L2 with a single premise:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,\wedge\mathrm{R} \qquad \frac{\Gamma, A \vdash C}{\Gamma, A \wedge B \vdash C}\,\wedge\mathrm{L1} \qquad \frac{\Gamma, B \vdash C}{\Gamma, A \wedge B \vdash C}\,\wedge\mathrm{L2}$$

In both classical and intuitionistic logic, due to the presence of weakening and contraction, the multiplicative and the additive presentations of the connective $\wedge$ are equivalent, and similarly for $\vee$. For example, in intuitionistic logic the multiplicative formulation of $\wedge$R can be derived from the additive one, and vice versa:

$$\frac{\dfrac{\Gamma \vdash A}{\Gamma, \Delta \vdash A}\,\mathit{weak} \qquad \dfrac{\Delta \vdash B}{\Gamma, \Delta \vdash B}\,\mathit{weak}}{\Gamma, \Delta \vdash A \wedge B}\,\wedge\mathrm{R} \qquad \frac{\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma, \Gamma \vdash A \wedge B}\,\wedge\mathrm{R}}{\Gamma \vdash A \wedge B}\,\mathit{contr}$$

Since Linear Logic lacks the unrestricted structural rules, the above equivalence is no longer provable. This means that the conjunction $\wedge$ defined by the multiplicative rules is actually different from the one introduced by the additive rules, and similarly for $\vee$.

$$\frac{}{A \vdash A}\,ax \qquad \frac{\Gamma \vdash A \qquad \Delta, A \vdash C}{\Gamma, \Delta \vdash C}\,cut$$

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

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

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

$$\frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\,p \qquad \frac{\Gamma, A \vdash C}{\Gamma, !A \vdash C}\,d \qquad \frac{\Gamma \vdash C}{\Gamma, !A \vdash C}\,w \qquad \frac{\Gamma, !A, !A \vdash C}{\Gamma, !A \vdash C}\,c$$

$$\frac{\Gamma \vdash A\langle \gamma/\alpha \rangle \qquad \gamma \notin FV(\Gamma)}{\Gamma \vdash \forall \alpha. A}\,\forall\mathrm{R} \qquad \frac{\Gamma, A\langle B/\alpha \rangle \vdash C}{\Gamma, \forall \alpha. A \vdash C}\,\forall\mathrm{L}$$

Figure 2.1: The logical system ILL_2.

2.2.2 Linear Logic and its fragments

We present Second-Order Intuitionistic Linear Logic (ILL_2), omitting for the sake of simplicity the connective "$\oplus$", its unit "0", and the existential quantifier "$\exists$".

- Its formulas are generated by the following grammar:

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

  where α is taken from a denumerable set of propositional variables and "$\forall$" is called *(second-order) universal quantifier.*

- The logical system ILL_2 (*Second-Order Intuitionistic Linear Logic*) is in Figure 2.1. It derives *sequents* $\Gamma \vdash A$, where Γ is a context (a multiset of formulas) and A is a formula. We recall that, given a context $\Gamma = A_1, \ldots, A_n$, $!\Gamma$ stands for $!A_1, \ldots, !A_n$, $FV(\Gamma)$ denotes the set of all free propositional variables in Γ, and $A\langle B/\alpha \rangle$ denotes the standard meta-level substitution of B for every occurrence α in A.

- The subsystem ILL (*Intuitionistic Linear Logic*) is obtained from ILL_2 by forgetting the clause for $\forall$ in the grammar of formulas and the inference rules $\forall$R and $\forall$L in Figure 2.1.

- The notion of cut-free derivation and the cut-elimination steps for ILL_2 are standard and both cut-elimination and confluence hold for it [90].

It is often useful to study subsystems of ILL_2 that are expressive enough to illustrate particular features. We shall call a *fragment* of ILL_2 a system obtained by ruling out some clauses from the language of formulas and the corresponding inference rules from Figure 2.1. Fragments can be divided into *propositional*, i.e. those ones which are free from $\forall$, and *second-order*, i.e. those including $\forall$. We shall consider both kinds of fragment as free from $\otimes$ and units: these can be added to propositional fragments, but in second-order fragments they are definable, so they can be neglected (this is the reason why we have parenthesis in the table below).

Let us introduce the most interesting fragments of ILL_2:

	$\multimap$	$\otimes$	&	!	$\forall$
IMLL	✓	(✓)			
IMALL	✓	(✓)	✓		
IMELL	✓	(✓)		✓	
IMLL_2	✓	(✓)			✓
IMALL_2	✓	(✓)	✓		✓
IMELL_2	✓	(✓)		✓	✓

The complete names of these fragments are given below:

- IMLL (resp. IMLL_2) is *Propositional* (resp. *Second-Order*) *Intuitionistic Multiplicative Linear Logic*;
- IMALL (resp. IMALL_2) is *Propositional* (resp. *Second-Order*) *Intuitionistic Multiplicative Additive Linear Logic*;
- IMELL (resp. IMELL_2) is *Propositional* (resp. *Second-Order*) *Intuitionistic Multiplicative Exponential Linear Logic.*

2.2.3 Simpson's Linear Lambda Calculus

The *Linear Lambda Calculus* $\Lambda^!$ is an untyped term calculus closely related to linear logic introduced by Simpson in [85]. In this calculus the usual λ-abstraction is expressed by two distinct operators: a linear abstraction $\lambda x.M$ and non-linear abstraction $\lambda !x.M$, which allows duplications of the argument. In particular, the argument of $\lambda !x.M$ is required to be suspended as thunk $!N$, which corresponds to the !-box of linear logic proof nets (see [42]).

The terms of the Linear Lambda Calculus are generated by the following grammar:

$$M := \ x \mid \lambda x.M \mid \lambda !x.M \mid MM \mid !M$$

where x is taken from a denumerable set of variables and x is bound in both $\lambda x.M$ and $\lambda !x.M$. We say that x is *linear in M* if x occurs free exactly once in M and, moreover, this free occurrence of x does not lie within the scope of a !-operator in M. A term M is said *linear* if, for every subterm of M of the form $\lambda x.N$, it holds that x is linear in N.

A version of β-reduction can be defined for $\Lambda^!$, and is called *surface reduction.* Surface reduction forbids evaluation under the scope of a !-operator, which reflects the idea that thunks are suspended computations. This requires the notion of *surface context*, i.e. a term in $\Lambda^!$ containing a unique hole $[\cdot]$, generated by the following grammar:

$$\mathcal{C} := [\cdot] \mid \lambda x.\mathcal{C} \mid \lambda !x.\mathcal{C} \mid \mathcal{C}M \mid M\mathcal{C}.$$

if $\mathcal{C}$ is a surface context and M is a term, then $\mathcal{C}[M]$ denotes the term obtained by substituting the unique hole in $\mathcal{C}$ with M allowing the possible capture of free variables in M.

Surface reduction is defined by the binary relation $\to_!$ on $\Lambda^!$ expressing a single step of reduction and given by the following rules:

$$(\lambda x.M)N \to_! M[N/x]$$
$$(\lambda x!.M)(!N) \to_! M[N/x].$$

that can apply to any surface context. Its reflexive and transitive closure is $\to_!^*$. A term is in (or is a) *surface normal form* if no surface reduction applies to it.

One can easily check that linearity is preserved under surface reduction, that is, if $M \to_! M'$ and M linear, then M' is linear. Moreover, if $M \in \Lambda^!$ is linear:

- *Confluence*: $M \to_!^* M_1$ and $M \to_!^* M_2$ imply that there exists N such that $M_1 \to_!^* N$ and $M_2 \to_!^* N$;
- *Uniformity*: whenever a k-step reduction sequence $M \to_!^* S$ exists, where S is a surface normal form, then all reductions from M reach S in exactly k steps. In particular, if M is normalizing under surface reduction then it is strongly normalizing.

2.3 Implicit Computational Complexity

2.3.1 Turing Machines

Turing Machines represent a model of computation introduced by Alan Mathison Turing in [91]. They can be thought of as finite state machines operating on an infinitely long tape divided into cells. Each cell contains exactly one symbol from a finite alphabet Γ endowed with two special symbols, the "blank" $\Box$ (the only symbol that can occur infinitely many times on the tape) and the "start" $\triangleright$ (that identifies the portion of the tape where computation takes place). The tape is equipped with a *head* that can potentially read or write symbols in the tape, one cell at a time. Based on the symbol the head is currently reading and the current state, the Turing Machine (over)writes a new symbol in the same cell (possibly the same as the previous one), moves left or right, and enters a new state (possibly the same as the previous one). The *transition function* is the "program" that specifies each of these actions.

More formally, a *(deterministic) Turing Machine*, TM for short, is a tuple $\mathcal{M} \triangleq (\Gamma, Q, \delta)$, where:

- Γ is a finite set of symbols, called the *alphabet* of $\mathcal{M}$, containing the symbols $\Box$ (*blank*) and $\triangleright$ (*start*);
- Q is a finite set, called the *set of states* of $\mathcal{M}$, containing a special state q_0 (*initial state*), and a subset F of distinguished states (*final states*);
- δ is a function $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\text{left}, \text{right}\}$, called *transition function of* $\mathcal{M}$.

The tape contains initially a finite string $\triangleright x_0 \dots x_n$ called *input*, with $x_i \in \Gamma \setminus \{\Box, \triangleright\}$, and blank symbols on the rest of its cells, with the head reading x_0 and the machine in the initial state q_0. The triple $(\triangleright, x_0 \dots x_n, q_0)$ describes this initial step, and is called *initial configuration.*

More in general, a *configuration* is a triple $(\triangleright x_0 \dots x_n, y_0 \dots y_m, q)$, with $x_i, y_j \in \Gamma \setminus \{\Box, \triangleright\}$ and $q \in Q$, describing a step in the computation of the machine: the string $\triangleright x_0 \dots x_n, y_0 \dots y_m$ represents all non-blank symbols that are currently written on the tape (there are finitely many such symbols), the head is reading the symbol y_0, and the machine is in the state q.

At each step in the computation, the machine applies the transition function δ to the pair $(q, x) \in Q \times \Gamma$ describing the current state and the symbol red by the head, and obtains a triple

(q', y, m), with $m \in \{\text{left}, \text{right}\}$, that provides the new state q' of the machine, the new symbol y to be written, and the move m of the head.

If the machine is in a final state, it halts. In this case, the string $\triangleright y_0 \dots y_n$ of non blank symbols in the tape is called *output*, and the corresponding configuration is called *final*. W.l.o.g. the set of final states of a TM can be divided into accepting and rejecting.

2.3.2 Computational Complexity

We here recall some basic definitions about computational complexity from Arora and Barak [5].

A *complexity class* is a set of *languages*, i.e. subsets of $\{0, 1\}^*$, that can be recognised by a Turing Machine $\mathcal{M}$ within given resource bounds (in time or space). More formally, given a language L and a function $T : \mathbb{N} \longrightarrow \mathbb{N}$, we say that $\mathcal{M}$ *recognises* L *in* $T(n)$*-time* (resp. *in* $T(n)$*-space*) if, for every $x \in \{0, 1\}^*$, whenever $\mathcal{M}$ is initialized to the initial configuration on input x:

- if $x \in L$ then $\mathcal{M}$ halts on an accepting state,
- if $x \notin L$ then $\mathcal{M}$ halts on a rejecting state,
- $\mathcal{M}$ requires at most $T(|x|)$ steps of computation (resp. cells of the tape).

We briefly recall some of the most interesting (deterministic) complexity classes:

- PTIME is the set of all languages that can be recognised by a TM in $p(n)$-time, for some polynomial $p : \mathbb{N} \longrightarrow \mathbb{N}$;
- PSPACE is the set of all languages that can be recognised by a TM in $p(n)$-space, for some polynomial $p : \mathbb{N} \longrightarrow \mathbb{N}$;
- EXPTIME is the set of all languages that can be recognised by a TM in $2^{p(n)}$-time, for some polynomial $p : \mathbb{N} \longrightarrow \mathbb{N}$;
- ELEMENTARY is the set of all languages that can be recognised by a TM in $\exp_k(n)$-time, for some $k \in \mathbb{N}$, where $\exp_0(n) = n$ and $\exp_{k+1}(n) = 2^{\exp_k(n)}$.

The class PTIME plays a central role in computational complexity. On the one hand, it is invariant for all models of computation that are polynomially equivalent to the (deterministic single-tape) TMs. On the other hand, it roughly corresponds to the class of problems that can be "efficiently" decided, and hence realistically solvable on a computer.

Its functional version is FPTIME, defined as the set of all functions $f : \{0, 1\}^* \longrightarrow \{0, 1\}^*$ that can be computed by a TM in $p(n)$-time, for some polynomial $p : \mathbb{N} \longrightarrow \mathbb{N}$.

2.3.3 Implicit Computational Complexity

Implicit Computational Complexity (ICC) is a subfield of computational complexity theory originated in the 1990s that aims at characterizing complexity classes without referring to machine models (e.g. Turing Machines, Register Machines, . . .), explicit bounds on computational resources (in time and space) or particular interpretations, but only by considering language restrictions or logical/computational principles entailing complexity properties.

ICC has borrowed techniques and results from several areas, like recursion theory, type theory and proof theory, paying a special attention to the polynomial time functions/problems. The development of ICC has been spurred by Cobham's 1965 work [21], in which a subclass of the primitive recursive functions has been proven to capture FPTIME. This subclass includes certain initial functions and is closed under composition and a restricted version of the recursion scheme, called *bounded recursion on notation.*

Cobham's result has yielded fruitful applications, that helped in understanding the polynomial time. However, it cannot be considered as an "implicit" approach to complexity, because it makes use of external bounds. The first recursion-theoretic characterizations of FPTIME in the style of ICC are due to Bellantoni and Cook [13] and Leivant [59]. In the former, the arguments of a $n+m$-ary function $f^{n+m} : \mathbb{N}^{n+m} \to \mathbb{N}$ are partitioned into "normal" and "safe" by means of a semicolumn:

$$f(x_1, \dots, x_n; y_{n+1}, \dots, y_{n+m})$$

the idea is that the recursive call of f can only appear in the "safe zone". This approach is called *safe recursion.* Leivant's result is rather based on the so-called *predicative* (or *ramified*) *recurrence*, in which each argument of a function comes with a "tier", keeping track of the recursive calls.

Though apparently different, the two proposals capture in essence the same intuition: it is forbidden to iterate a function which is itself defined by recursion. More formally, in a recursive definition $f(s(x), y) = h(x, y, f(x, y))$ the result $f(x, y)$ of the recursive call of f cannot be itself a recursive parameter of the step function h. This idea is part of a more general approach at the very heart of ICC: *stratification.* Intuitively, stratification consists in organizing computation into different levels (e.g. Leivant's tiers) in order to cut off all those computations overstepping the given resource bounds.

The ramified recurrence has been generalized to higher-order languages, where functions can take other functions as argument. In this setting, the polynomial time is captured by imposing linearity constraints on higher order variables, like in Hofmann [49] with the system SLR (Safe Linear Recursion), or in Bellantoni et al. [14] with the calculus RA. In SLR, a modality $\Box$ provides the type-theoretical counterpart of Bellantoni and Cook's first-order notion of "normal" parameter: the type $(\Box\mathbf{N})^n \to \mathbf{N}^m \to \mathbf{N}$ corresponds exactly to a number-theoretic function with n normal variables and m safe variables.

The system SLR has inspired several ICC-like approaches to non-deterministic and probabilistic polytime complexity. First, Mitchell et al. [72] introduced OSLR, a variant of SLR that characterizes oracle polynomial time functionals. Secondly, Zhang [95] introduced another version of the system, called CSLR, that defines exactly those functions that can be computed by a Probabilistic Turing Machine in polynomial time. In SLR and all these variants, the characterization theorem exploits semantic methods, due to problems with the Subject reduction. To circumvent this technical drawback, Dal Lago and Toldin devised in [28] a further variation of Hofmann's system, called RSLR. This system has been proven both sound and complete for the probabilistic polynomial time by purely syntactic means.

A proof-theoretical approach to ICC relies on the so-called *light logics*, subsystems of Linear Logic obtained by considering weaker versions of the exponential rules able to limit the use of duplication and to induce a bound on normalization. In a light logic stratification is given by the notion of "depth", that allows a level-by-level cut-elimination.

A first example of light logic is Light Linear Logic (LLL), introduced by Girard in 1998 [44]. This logic and its variant Light Affine Logic (LAL) [6, 7] have been proven to capture FPTIME. Another characterization of the polynomial time has been proposed by Lafont with Soft Linear Logic (SLL) [56]. In this logic, contraction, weakening and dereliction are replaced by the so-called "multiplexor rule", that can be used to keep track of the number of duplications performed during cut-elimination in a very simple way. Last, Danos and Joinet designed Elementary Linear Logic (ELL) in [30], a light logic characterizing the class ELEMENTARY, containing all those languages that can be computed in time bounded by a tower of exponentials of fixed length.

Light logics were also studied as type assignments for the standard λ-calculus, like Baillot and Terui [88] for LAL or Gaboardi and Ronchi [38, 40] for SLL. According to the well-known Curry-Howard isomorphisms paradigm, this essentially amounts to decorate logical derivations with λ-terms. In these works it has been stressed that the modal aspect of light logics prevents a close correspondence between proof normalization and β-reduction: on the one hand, as also remarked in a more general setting by Lincoln [62], the subject reduction property fails, so that typed terms might become untypable during evaluation; on the other hand, the proof-theoretic complexity bounds on the cut-elimination are not always inherited by terms. In order to recover the missing matching, variants of these systems have been considered, like DLAL (Dual Light Affine Logic) in [88] and STA (Soft Type Assignment) in [38, 40].

2.4 Bisimulation and coinduction

Bisimulation is a fundamental notion in computer science (see e.g. [81]). It has been employed in several areas, like formal verification or program analysis, introducing new proof techniques based on coinduction. Moreover, the bisimulation equality, called *bisimilarity*, is the most studied form of behavioural equivalence in concurrency theory, being accepted as the *finest equivalence* we are willing to impose on processes.

Bisimulation is in general presented as a relation on state transition systems. A *Labelled Transition System*, *LTS* for short, is a triple $(W, Act, \{\xrightarrow{a}\}_{a\in Act})$, where W is a set of *states*, Act is a set of *labels* and the $\xrightarrow{a}$ are binary relations on W called *transition relations*. A *bisimulation* is a relation $\mathcal{R}$ on W such that, whenever $s_1 \mathrel{\mathcal{R}} s_2$:

(1) for all s_1' with $s_1 \xrightarrow{a} s_1'$, there is s_2' such that $s_2 \xrightarrow{a} s_2'$ and $s_1' \mathrel{\mathcal{R}} s_2'$:

$$\begin{tikzcd} s_1 \arrow[r, "a"] & s_1' \\ \mathcal{R} & \mathcal{R} \\ s_2 \arrow[r, "a"] & \exists s_2' \end{tikzcd}$$

(2) for all s_2' with $s_2 \xrightarrow{a} s_2'$, there is s_1' such that $s_1 \xrightarrow{a} s_1'$ and $s_1' \mathrel{\mathcal{R}} s_2'$:

$$\begin{tikzcd} s_1 \arrow[r, "a"] & \exists s_1' \\ \mathcal{R} & \mathcal{R} \\ s_2 \arrow[r, "a"] & s_2' \end{tikzcd}$$

if $\mathcal{R}$ satisfies only point (1), then it is called a *simulation*. *Bisimilarity*, written $\sim$, is the union of all bisimulations, thus $s \sim t$ holds if there exists a bisimulation $s \mathrel{\mathcal{R}} t$. *Similarity*, written $\precsim$, is the union of all simulations.

Let us remark that bisimilarity has a strong impredicative flavour:

- bisimilarity is in turn a bisimulation, hence it is contained in the union of all relations defining it;
- in order to show that s_1 and s_2 are bisimilar it suffices to find a bisimulation that contains the pair (s_1, s_2) (a technique known as *bisimulation proof method*).

The impredicative aspect of bisimilarity is a consequence of its coinductive nature. Intuitively, a set A is defined *coinductively* if it is the *greatest* solution of an inequation of a certain form, and the related *coinduction proof principle* just says that any set that is a solution of the same inequation *is contained in* A. Dually, a set A is defined *inductively* if it is the *least* solution of an inequation of a certain form, and the related *induction proof principle* says that any other set that is solution of the same inequation *contains* A.

Showing that bisimilarity and the related bisimulation proof method are based on coinduction relies on a simple application of the Knaster-Tarski Theorem concerning complete lattices. Before stating it, we recall that a *complete lattice* is a partially ordered set with all joins (i.e. all subsets have a least upper bound) and meets (i.e. all subets have an greatest lower bound). If we denote $\leq$ the partial order on a lattice L, a point x in the lattice is a *post-fixed point* of an endofunction F on L if $x \leq F(x)$; it is a *pre-fixed point* if $F(x) \leq x$; finally, it is a *fixed-point* if it is both a post-fixed and a pre-fixed point, i.e. if $F(x) = x$.

Theorem 1 (Knaster-Tarski)**.** *On a complete lattice, a monotone endofunction has a complete lattice of fixed points. In particular, the greatest fixed point of the function is the join of all its post-fixed points, and the least fixed point is the meet of all its pre-fixed points.*

The existence of the gratest fixed point for monotone functions justifies coinductive definitions and allows the coinduction proof principle expressed by the following rule:

$$\frac{F \text{ monotone} \qquad x \leq F(x)}{x \leq \mathrm{gfp}(F)}$$

where $\mathrm{gfp}(F)$ indicates the greatest fixed-point of F.

Now, given a LTS $(W, Act, \{\xrightarrow{a}\}_{a \in Act})$ we can consider the complete lattice of all relations on W partially ordered by the inclusion, where the join is the relational union and the meet is the relational intersection. Moreover, on this lattice we consider the endofunction $F_\sim$ such that, for all relations $\mathcal{R}$ on W, $F_\sim(\mathcal{R})$ is the greatest bisimulation contained in $\mathcal{R}$.

The function $F_\sim$ satisfies the following properties:

(1) $F_\sim$ is monotone;

(2) $\mathcal{R}$ is a bisimulation if and only if $\mathcal{R} \subseteq F_\sim(\mathcal{R})$.

Point (1) and Theorem 1 imply the existence of a greatest fixed-point $\mathcal{R}^*$ for $F_\sim$, which is the union of all its post-fixed point, i.e. the union of all relations $\mathcal{R}$ such that $\mathcal{R} \subseteq F_\sim(\mathcal{R})$. By point (2), we conclude $\mathcal{R}^* = \sim$.

2.5 Notational conventions and basic definitions

2.5.1 Standard notation

We write $\mathbb{N}$ for the set of natural numbers, $\mathbb{R}$ for the set of real numbers and $[0, 1]$ for the unit interval of $\mathbb{R}$. Given a finite set X, a *string (over X)* is a finite ordered tuple of elements in X. We denote X^n the set of strings over X of length n (X^0 being the singleton consisting of the empty tuple), and X^* is defined by $\bigcup_{n \in \mathbb{N}} X^n$. The length of a string x is denoted $|x|$. We will typically consider strings over the binary alphabet $\{0, 1\}$, also called *strings of booleans.* Elements of $\{0, 1\}$ are ranged over by b.

2.5.2 Relations and distributions

Relations and inductive presentation. We assume the reader is familiar with standard set-theoretic notions (see [55]). We recall that, if $X_1, \ldots, X_n$ are sets, a *relation on* $X_1, \ldots, X_n$ is a subset of $X_1 \times \ldots \times X_n$. A *relation over* X is a relation on X, X. Relations are ranged over by $\mathcal{R}$. Special relations will be denoted by special symbols like, for example, "$\Rightarrow$" (Chapter 4) and "$\Downarrow$" (Chapter 5). For binary relations, we shall use the standard infix notation $x\ \mathcal{R}\ y$ for $(x, y) \in \mathcal{R}$. Given a relation on X, Y and $Z \subseteq X$, $\mathcal{R}(Z)$ denotes the image of Z under $\mathcal{R}$, i.e. the set $\{y \in Y \mid \exists x \in Z,\ (x, y) \in \mathcal{R}\}$, and $\mathcal{R}^{op}$ represents the converse of $\mathcal{R}$, i.e. $\{(y, x) \in Y \times X \mid (x, y) \in \mathcal{R}\}$. Moreover, given a relation over X, $\mathcal{R}^+$ (resp. $\mathcal{R}^*$) denotes the transitive (resp. reflexive and transitive) closure of $\mathcal{R}$. Finally, if $\mathcal{R}$ is an equivalence relation, $X/\mathcal{R}$ denotes the set of all equivalence classes of X modulo $\mathcal{R}$.

Binary relations can be introduced by means of an inductive, rule-based definition (see [3]). In this case, we display a set of rule schemes $\{R_1, \ldots, R_n\}$ defining a relation $\mathcal{R}$, where each R_i has the following form:

$$\frac{x_1\ \mathcal{R}\ y_1 \ldots x_n\ \mathcal{R}\ y_n}{x\ \mathcal{R}\ y}\ R_i$$

with possible side conditions that narrow the premises of R_i. Given a system of rule schemes $\{R_1, \ldots, R_n\}$ defining a binary relation $\mathcal{R}$, a derivation of $x\ \mathcal{R}\ y$ is a downward oriented tree whose nodes are rules (the leaves being the axioms, i.e. rules with an empty set of premises), whose edges $R \to R'$ are such that the conclusion of the rule R is a premise of the rule R', and whose root is a rule with conclusion $x\ \mathcal{R}\ y$. Derivations are ranged over by Greek letters like π, ρ, σ. With $\pi : x\ \mathcal{R}\ y$ we denote a derivation π of $x\ \mathcal{R}\ y$.

Probabilistic transition relations. For relations $\mathcal{R} \subseteq X \times [0, 1] \times X$ we shall use the infix notation $x\ \mathcal{R}_r\ y$ in place of $(x, r, y) \in \mathcal{R}$. A *probabilistic transition relation (over a set X)* is a relation $\mathcal{R} \subseteq X \times [0, 1] \times X$ such that, for all $x \in X$:

$$\sum_{\substack{r,\, y \text{ s.t.} \\ x\ \mathcal{R}_r\ y}} r \leq 1$$

Given $\mathcal{R}$ a probabilistic transition relation over X, we construct by induction on $n \in \mathbb{N}$ the probabilistic transition relation $\mathcal{R}^n$:

$$\begin{aligned} x\ \mathcal{R}^0_r\ y &\Leftrightarrow x = y\ \wedge\ r = 1 \\ x\ \mathcal{R}^{n+1}_r\ y &\Leftrightarrow \exists y'\, \exists r'\, \exists r''\, (x\ \mathcal{R}^n_{r'}\ y'\ \wedge\ y'\ \mathcal{R}_{r''}\ y\ \wedge\ r = r' \cdot r''). \end{aligned}$$

Subprobability distributions. Let X be a set. A *subprobability distribution (over X)* is a function $f : X \to [0, 1]$ such that $\sum_{x \in X} f(x) \leq 1$. If $\sum_{x \in X} f(x) = 1$, then f is also called a *probability distribution (over X)*.

The set of all subprobability distributions over X is denoted by $\mathfrak{D}(X)$. Subprobability distributions, or distributions for short, are ranged over by $\mathscr{D}, \mathscr{E}, \mathscr{F}, \ldots$. Given a distribution $\mathscr{D} \in \mathfrak{D}(X)$, its *support* $\mathrm{supp}(\mathscr{D})$ is the subset of all elements $x \in X$ such that $\mathscr{D}(x) > 0$. Given $x_1, \ldots, x_n \in X$, the expression $r_1 \cdot x_1 + \ldots + r_n \cdot x_n$ is used to denote the distribution $\mathscr{D} \in \mathfrak{D}(X)$ with finite support $\{x_1, \ldots, x_n\}$ such that $\mathscr{D}(x_i) = r_i$, for every $i \leq n$. Given $Y \subseteq X$ and $\mathscr{D} \in \mathfrak{D}(X)$, we write $\mathscr{D}(Y)$ in place of $\sum_{x \in Y} \mathscr{D}(x)$. In particular, $\mathscr{D}(X)$ will be also written $\sum \mathscr{D}$, and called the *mass* of $\mathscr{D}$. The symbol $\perp$ denotes the empty distribution and x can denote both an element in X and the distribution having all its mass on x. Given a (possibly infinite) index set I, a family $\{r_i\}_{i\in I}$ of positive real numbers such that $\sum_{i\in I} r_i \leq 1$, and a family $\{\mathscr{D}_i\}_{i\in I}$ of distributions, the distribution $\sum_{i\in I} r_i \cdot \mathscr{D}_i$ is defined, for all $x \in X$, by $(\sum_{i\in I} r_i \cdot \mathscr{D}_i)(x) = \sum_{i\in I} r_i \cdot \mathscr{D}_i(x)$.

Subprobability distributions over a set X can be compared by pointwise lifting the canonical order on $\mathbb{R}$ up to $\mathfrak{D}(X)$. For all $\mathscr{D}, \mathscr{E} \in \mathfrak{D}(X)$:

$$\mathscr{D} \leq_{\mathfrak{D}} \mathscr{E} \text{ if and only if } \forall x \in X,\ \mathscr{D}(x) \leq \mathscr{E}(x). \tag{2.1}$$

We recall that, given a partial order set $(X, \sqsubseteq)$, a subset $\emptyset \neq D \subseteq X$ is *directed* if $\forall x, y \in D$ $\exists z \in D$ such that $x \sqsubseteq z$ and $y \sqsubseteq z$. A *directed-complete partial order* (or *dcpo*) is a partial order in which every directed set D has a supremum $\bigsqcup D$.

For any set X, it holds that:

$$(\mathfrak{D}(X), \leq_{\mathfrak{D}}) \text{ is a dcpo} \tag{2.2}$$

where the least element is the empty distribution $\perp$ and the supremum of a directed subset $\emptyset \neq D \subseteq \mathfrak{D}(X)$ is pointwise defined, for all $x \in X$, by $(\sup_{\mathscr{D}\in D} \mathscr{D})(x) \triangleq \sup_{\mathscr{D}\in D} \mathscr{D}(x)$.

2.5.3 Typed and untyped calculi

The standard λ-calculus. We assume the reader is familiar with some basic concepts related to the standard λ-calculus (see [10]) like: (i) the set $FV(M)$ of the free variables of the λ-term M, where M is *closed* if $FV(M) = \emptyset$, (ii) the meta-level *substitution* $M[N/x]$ that replaces the λ-term N for every free occurrence of the variable x in M, (iii) the *size* $|M|$ of a λ-term M, i.e. the number of nodes in its syntax tree, (iv) the *contexts* $\mathcal{C}$, i.e. λ-terms with a place-holder (the hole) $[\cdot]$ that may capture free variables of a λ-term plugged into $[\cdot]$, (v) the *α-equivalence* $(=_\alpha)$, (vi) the *β-reduction* $(\lambda x.M)N \to_\beta M[N/x]$, (vii) the *$\eta$-reduction* $\lambda x.Mx \to_\eta M$ that can be applied if $x \notin FV(M)$.

Terms will be taken up to α-equivalence, and both $\to_\beta$ and $\to_\eta$ will be considered contextually closed. By $\to^*_\beta$ we denote the reflexive and transitive closure of the β-reduction, and by $=_\beta$ its reflexive, symmetric and transitive closure. Also, by $\to^*_\eta$ we denote the reflexive and transitive closure of the η-reduction, and by $=_\eta$ its reflexive, symmetric and transitive closure. Finally, by $\to_{\beta\eta}$ we denote $\to_\beta \cup \to_\eta$, and by $\to^*_{\beta\eta}$ we denote its reflexive and transitive closure.

We recall that a λ-term is in *β-normal form*, or simply (β-)*normal*, whenever no β-reduction applies to it. As a consequence, all β-normal forms have shape $\lambda x_1 \ldots x_n.yM_1 \ldots M_m$, where each M_i is in turn a β-normal form, for $n, m \geq 0$. A λ-term is in *η-normal form*, or simply *η-normal*, if no η-reduction applies to it. Finally, a λ-term is in *$\beta\eta$-normal form*, or simply *$\beta\eta$-normal*, whenever no $\beta\eta$-reduction applies to it.

Type systems. In Section 2.1 and Section 2.2 we introduced some basic notions of proof-theory (see [90]), Linear Logic (see [46]), type systems (see [11]) and the Curry-Howard isomorphisms (see [87]). We shall mainly consider type systems that correspond, under the Curry-Howard paradigm, to second-order fragments of ILL_2 or to their proper extensions.

We recall that type systems derive *judgments* $\Gamma \vdash M : A$, where A is a type (the *predicate*), M is a term (the *subject*), and Γ is a *context*, i.e. a finite multiset of the form $x_1 : A_1, \ldots, x_n : A_n$, where each x_i is a λ-variable, each A_i is a type, and each $x_i : A_i$ is called *declaration*. Typically, names for contexts are Γ, Δ or Σ. Given a context Γ, its *domain* $\text{dom}(\Gamma)$ is $\{x_1, \ldots, x_n\}$ and its range $\text{rng}(\Gamma)$ is $\{A_1, \ldots, A_n\}$.

The set of free type variables of A will be denoted by $FV(A)$. If $FV(A) = \emptyset$, then A is said *closed*. If $FV(A) = \{\alpha_1, \ldots, \alpha_n\}$, then *a closure* $\overline{A}$ of A is $\forall\alpha_1. \cdots .\forall\alpha_n.A$, not necessarily linked to a specific order of $\alpha_1, \ldots, \alpha_n$. The standard meta-level substitution of a type B for every free occurrence of α in A is $A\langle B/\alpha\rangle$. The *size* $|A|$ of the type A is the number of nodes in its syntax tree. Given $\Gamma = x_1 : A_1, \dots, x_n : A_n$, we shall write $FV(\Gamma)$ in place of $\bigcup_{A \in \mathrm{rng}(\Gamma)} FV(A)$, and $|\Gamma|$ in place of $\sum_{i=1}^{n} |A_i|$.

Derivations in a type system are ranged over by $\mathcal{D}$. The *size* $|\mathcal{D}|$ of $\mathcal{D}$ is the number of rule instances $\mathcal{D}$ contains. We say that $\Gamma \vdash M : B$ is *derivable* if a derivation $\mathcal{D}$ exists that concludes with the judgment $\Gamma \vdash M : B$, and we also say that $\mathcal{D}$ is a derivation of $\Gamma \vdash M : B$. In that case we write $\mathcal{D} \triangleleft \Gamma \vdash M : B$ saying that M is an *inhabitant* of B or that B is *inhabited* by M from Γ.

Chapter 3

A Type Assignment of Linear Erasure and Duplication

Through the well-known Curry-Howard isomorphisms, that connect logical proofs and functional languages, Intuitionistic Linear Logic (ILL) allows to look at computation as the result of an interaction between non-linear components, where arguments can be duplicated and erased at will, and strictly linear components, where each argument is consumed exactly once. In the standard λ-calculus, seen as a paradigm of functional programming, the latter components of computation are represented by the *linear λ-terms.*

According to the above computational reading, the core fragment of ILL forbidding weakening and contraction (i.e. IMLL) and its second-order formulation (i.e. IMLL_2) are both seen as type systems for the linear λ-terms: formulas are types, proofs are terms, and cut-elimination is term reduction. It becomes then possible to explore the computational expressiveness of these logics which, despite a very weak language, has been proven quite rich.

A first result in this direction has been given by Alves et al. [4]. They have shown that extending the language of IMLL with booleans, natural numbers, and a linear iterator is enough to obtain the full computational power of Gödel's system $\mathcal{T}$. In other words, the presence of explicit rules for weakening and contraction is redundant in system $\mathcal{T}$, since non-linearity can be recovered from numerals and linear iteration.

The work of Alves et al. is based on an extension of IMLL. In [68] Matsuoka investigates the discriminating power of IMLL *without* new constructs. His main result is a typed variant of the *weak* Böhm Theorem: given a pair of distinct closed $\beta\eta$-normal forms N_1 and N_2 having type A in IMLL, a term M exists such that MN_1 is $\beta\eta$-equivalent to $\mathtt{tt}$ and MN_2 is $\beta\eta$-equivalent $\mathtt{ff}$, where $\mathtt{tt}$ and $\mathtt{ff}$ are suitable encodings of booleans. This result has been later refined by the same author in [69], who proves a typed version of the *strong* Böhm Theorem, generalizing $\mathtt{tt}$ and $\mathtt{ff}$ to arbitrary closed terms of a given type in IMLL.

A remarkable consequence of [68, 69] is that discriminating among linear λ-terms in a typed setting relies on some *implicit erasure mechanisms* that can be expressed in IMLL, though this system lacks the inference rules for weakening and contraction. How is this possible? There are essentially two kinds of erasure mechansms in the linear λ-calculus. A first approach has been suggested by Klop [54], and consists on regarding some components of a term as "garbage". Linear erasure becomes then a form of garbage collection, in which all the undesired "data" produced during evaluation are moved to these components. A second approach is based on data consumption. According to this idea, erasing a term amounts to "fully evaluate" it by a sequence of applications until an identity $\mathbf{I} = \lambda x.x$ is reached. Both erasure mechanisms have been recently used by Mackie [63] to show some encodings of natural numbers and basic number-theoretic functions in the linear λ-calculus.

Special forms of linear duplication exist as well. In [64] Mairson has shown that a "duplicator" of booleans can be typed in IMLL, proving that the system is powerful enough to encode boolean circuits. Here duplication exploits linear erasure (by data consumption), and is constructed with built-in pairs of booleans $\langle \mathtt{tt}, \mathtt{tt} \rangle$ and $\langle \mathtt{ff}, \mathtt{ff} \rangle$ representing both possible outcomes of duplication: according to the boolean received in input, the duplicator selects the right pair and erases the remaining one.

The encoding of boolean circuits has been reformulated in IMLL_2 by Mairson and Terui [65], where the presence of second-order quantifiers plays a central role, because it allows to assign uniform types to structurally related linear λ-terms. Moreover, these authors generalized the mechanisms of linear erasure and duplication, known for booleans, to the so-called class of closed Π_1 types, representing finite data types. Showing the existence of a uniform duplication mechanism for all closed and normals inhabitants of a closed Π_1 type is not simple, and its proof was only sketched in [65]. It basically consists in constructing in IMLL_2 a "compiler", that maps each such inhabitant M into a linear λ-term $\lceil M \rceil$ representing its encoding, and a "decoder", that maps $\lceil M \rceil$ back to the desired pair $\langle M, M \rangle$.

Starting from [65], this chapter investigates the basic proof-theoretical and computational properties of *Linearly Exponential Multiplicative Type Assignment* (LEM), a new system able to internalize the linear weakening and contraction of closed Π_1 types (here called "ground types") discussed by Mairson and Terui. LEM extends IMLL_2 with inference rules for the modality "$\downarrow$" that recall the exponential rules of ILL, though much weaker.

To faithfully represent the mechanisms of linear erasure and duplication of IMLL_2, we introduce constrained forms of cut-elimination rules for LEM, we call "lazy". Lazy cut-elimination rules are far too weak for assuring a cut-elimination result. So, we identify a relevant class of types, called "lazy types", whose derivations can always be rewritten in cubic time to cut-free ones according to a specific lazy cut-elimination strategy. We also prove the Subject reduction property for LEM.

Then, we study a translation of LEM into IMLL_2, that shows how the restricted weakening and contraction rules of LEM can be represented by linear weakening and contractions of closed Π_1 types in IMLL_2. A striking feature of the translation is that contraction in LEM exponentially compresses the linear contraction of IMLL_2.

The translation reveals that the algorithmic expressiveness of LEM is the same as IMLL_2. Nevertheless, the exponential compression of the linear contraction in IMLL_2 produces several benefits in LEM. On the one hand, the encoding of boolean circuits for IMLL_2 can be converted into a more compact and modular one for LEM. On the other hand, we show a nice encoding of natural numbers quite similar to the Church encoding in Linear Logic, and we show that both the successor and the addition are definable.

Outline of the chapter. In this chapter we present the type system LEM, and we investigate its basic properties. In Section 3.1, we introduce the linear λ-calculus and its type assignment system IMLL_2 (Section 3.1.1) in order to explore the mechanisms of linear duplication and erasure both in the untyped and in the typed setting (Sections 3.1.2 and 3.1.3). In Section 3.2 we give a complete and detailed proof of the duplication theorem sketched in [65]. In Section 3.3 we present the type system LEM and we explore the proof-theoretical and the computational properties. First, we prove a mildly weakened form of cut-elimination, we call lazy (Section 3.3.2), and the Subject reduction property (Section 3.3.3). Then, we show a translation of LEM into IMLL_2, proving that derivations in the former system can exponentially compress the ones in the latter (Section 3.3.4). Finally, in Section 3.4 we explore the benefit of the exponential compression on the algorithmic expressiveness of LEM, by considering an encoding of boolean circuits (Section 3.4.1) and an encoding of the natural numbers that allows to represent the successor and the addition (Section 3.4.2).

3.1 Duplication and erasure in the linear λ-calculus

In this section we discuss the erasure and duplication in the linear λ-calculus Λ_l. Concerning the untyped case, we study the existence of the so-called "erasers" and "duplicators", i.e. linear λ-terms that uniformly erase or duplicate all elements of a subset of Λ_l. On the one hand, we prove that an eraser for a finite set $X \subseteq \Lambda_l$ exists if and only if all terms in X are closed (Proposition 5). On the other hand, we show that if a set $X \subseteq \Lambda_l$ has a duplicator, then X must be finite and contains only closed terms (Proposition 6). What about the converse? Is it the case that a duplicator exists for all finite subsets containing closed linear λ-terms? We argue that this property essentially relies on a conjecture about Böhm Separation (Conjecture 7).

In a typed setting, we focus on sets of closed and normal inhabitants having type in IMLL_2 (Second-Order Intuitionistic Multiplicative Linear Logic). Theorem 9 and Theorem 10 ensure the existence of duplicators and erasers for "ground types", i.e. Mairson and Terui's closed Π_1 types [65]. Constructing a "duplicator" for a ground type is not simple and requires several technical preliminary results. For this reason the detailed proof of Theorem 10 will be postponed to the next section.

3.1.1 The linear λ-calculus and IMLL_2

The *linear* λ-calculus is the λ-calculus restricted to *linear* λ-terms.

Definition 1 (Linear λ-terms)**.** A *linear λ-term* is a λ-term M such that:

- each free variable of M has just one occurrence free in it;
- for each subterm $\lambda x.N$ of M, x occurs in N exactly once.

The set of all linear λ-terms is denoted Λ_l, while its restriction to closed λ-terms is $\Lambda_l^{\emptyset}$.

Example 5. Examples of linear λ-terms are the identity $\mathbf{I} \triangleq \lambda x.x$ and the exchange operator $\mathbf{C} \triangleq \lambda x.\lambda y.\lambda z.xzy$, while the constant operator $\mathbf{K} \triangleq \lambda x.\lambda y.x$ and the (strong) composition operator $\mathbf{S} \triangleq \lambda x.\lambda y.\lambda z.xz(yz)$ are not linear λ-terms. Last, if M and N are linear λ-terms, then $\langle M, N\rangle \triangleq \lambda z.zMN$ is a linear λ-term.

Linear λ-terms enjoy the following straightforward properties:

Proposition 2. *For all $M \in \Lambda_l$, if $M \to_{\beta\eta} N$ then:*

(1) $FV(N) = FV(M)$,

(2) $N \in \Lambda_l$.

(3) $|M| = |N| + 3$.

Proof. Point (1) and point (2) are straightforward. Concerning point (3), each β-reduction $(\lambda x.M)N \to_\beta M[N/x]$ is such that $|(\lambda x.M)N| = |M| + |N| + 2$ and $|M[N/x]| = |M| + |N| - 1$, the latter because the unique occurrence of x in M has size 1 and is replaced by N. Moreover, in each η-reduction $\lambda x.Mx \to_\eta M$, we remove an abstraction, an application and a variable. $\square$

We present IMLL (Propositional Intuitionistic Multiplicative Linear Logic) and IMLL_2 (Second-Order Intuitionistic Multiplicative Linear Logic) as type assignment systems for the linear λ-calculus:

Definition 2 (The systems IMLL_2 and IMLL)**.**

- Let $\mathcal{X}$ be a denumerable set of variables. The set of *types* of IMLL_2 are generated by the following grammar:
$$A := \alpha \mid A \multimap A \mid \forall\alpha.A \tag{3.1}$$
where $\alpha \in \mathcal{X}$.
- IMLL_2 is the type assignment system for the linear λ-calculus displayed in Figure 3.1(a) (in sequent calculus style) and in Figure 3.1(b) (in natural deduction style), where the rules $\multimap$L and $\multimap$E enjoy the following *linearity constraint*: $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Delta) = \emptyset$.

The system IMLL is obtained from IMLL_2 by forgetting the clause $\forall\alpha.A$ in (3.1) and the inference rules for $\forall$ in Figure 3.1.

$$\dfrac{}{x : A \vdash x : A}\,ax \qquad \dfrac{\Gamma \vdash N : A \qquad \Delta, x : A \vdash M : C}{\Gamma, \Delta \vdash M[N/x] : C}\,cut$$

$$\dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B}\,{\multimap}\mathrm{R} \qquad \dfrac{\Gamma \vdash N : A \qquad \Delta, x : B \vdash M : C}{\Gamma, \Delta, y : A \multimap B \vdash M[yN/x] : C}\,{\multimap}\mathrm{L}$$

$$\dfrac{\Gamma \vdash M : A\langle\gamma/\alpha\rangle \qquad \gamma \notin FV(\Gamma)}{\Gamma \vdash M : \forall\alpha.A}\,\forall\mathrm{R} \qquad \dfrac{\Gamma, x : A\langle B/\alpha\rangle \vdash M : C}{\Gamma, x : \forall\alpha.A \vdash M : C}\,\forall\mathrm{L}$$

(a) IMLL_2 in sequent calculus style.

$$\dfrac{}{x : A \vdash x : A}\,ax$$

$$\dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B}\,{\multimap}\mathrm{I} \qquad \dfrac{\Gamma \vdash M : A \multimap B \qquad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B}\,{\multimap}\mathrm{E}$$

$$\dfrac{\Gamma \vdash M : A\langle\gamma/\alpha\rangle \qquad \gamma \notin FV(\Gamma)}{\Gamma \vdash M : \forall\alpha.A}\,\forall\mathrm{I} \qquad \dfrac{\Gamma \vdash M : \forall\alpha.A}{\Gamma \vdash M : A\langle B/\alpha\rangle}\,\forall\mathrm{E}$$

(b) IMLL_2 in natural deduction style.

Figure 3.1: IMLL_2 as a type assignment system.

The following is a well-known result from Hindley [48]:

Theorem 3 ([48])**.** *Every linear λ-term is typable in* IMLL.

As a consequence, IMLL_2 gives a type to every linear λ-term. The converse holds as well, due to the above *linearity constraint* in Definition 2, so the class of linear λ-terms is exactly the one of all typable λ-terms in IMLL and IMLL_2. It follows that second-order does not allow to type more terms but it is nevertheless useful to assign uniform types to structurally related λ-terms.

Tensor and unit are definable in IMLL_2 according to the following definition:

Definition 3 (Unit and tensor)**.** The *unit* and the *tensor* are definable in IMLL_2 as follows:

$$\begin{array}{ll}
\mathbf{1} \triangleq \forall\alpha.(\alpha \multimap \alpha) & A \otimes B \triangleq \forall\alpha.(A \multimap B \multimap \alpha) \multimap \alpha \\
\mathbf{I} \triangleq \lambda x.x & \langle M, N\rangle \triangleq \lambda z.z\,M\,N \\
\texttt{let } M \texttt{ be } \mathbf{I} \texttt{ in } N \triangleq MN & \texttt{let } M \texttt{ be } x, y \texttt{ in } N \triangleq M(\lambda x.\lambda y.N).
\end{array}$$

Both binary tensor products and pairs extend to their obvious n-ary versions $A^n = A \otimes .\overset{n}{\cdot}. \otimes A$ and $M^n \triangleq \langle M, .\overset{n}{\cdot}., M\rangle$.

Henceforth, any occurrence of unit, (n-ary) tensor and n-tuple will be taken from Definition 3. So, the reduction rules and the inference rules in Figure 3.2 will be considered as derivable in IMLL_2.

We conclude by stressing a fundamental difference between IMLL and IMLL_2. The latter allows types with infinite (closed and) normal inhabitants, as the following example shows:

Example 6. Consider the type $\mathbf{1} = \forall\alpha.(\alpha \multimap \alpha)$. For all $n \geq 1$ we can construct the following derivation of $\mathbf{1} \multimap \mathbf{1}$:

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathbf{I} : \mathbf{1}}\,ax \qquad \dfrac{}{v : \mathbf{1} \vdash v : \mathbf{1}}\,ax}{z : \mathbf{1} \multimap \mathbf{1} \vdash z\mathbf{I} : \mathbf{1}}\,{\multimap}\mathrm{L}}{z : \mathbf{1} \vdash z\mathbf{I} : \mathbf{1}}\,\forall\mathrm{L}$$

$$\vdots$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \mathbf{I} : \mathbf{1}}\,ax \qquad y : \mathbf{1} \vdash y\mathbf{I}.\overset{n-1}{\cdot}.\mathbf{I} : \mathbf{1}}{x : \mathbf{1} \multimap \mathbf{1} \vdash x\mathbf{I}.\overset{n}{\cdot}.\mathbf{I} : \mathbf{1}}\,{\multimap}\mathrm{L}}{x : \mathbf{1} \vdash x\mathbf{I}.\overset{n}{\cdot}.\mathbf{I} : \mathbf{1}}\,\forall\mathrm{L}}{\vdash \lambda x.x\mathbf{I}.\overset{n}{\cdot}.\mathbf{I} : \mathbf{1}}\,{\multimap}\mathrm{R}$$

All elements in $\{\lambda x.x\} \cup \{\lambda x.x\mathbf{I}.\overset{n}{\cdot}.\mathbf{I} \mid n \geq 1\}$ are (closed and) normal inhabitants of $\mathbf{1} \multimap \mathbf{1}$.

3.1.2 The untyped setting

The linear λ-calculus forbids any form of *direct* duplication of λ-terms, by means of multiple occurrences of the same variable, or of erasure, by omitting occurrences of bound variables in a λ-term. Nevertheless, erasure and duplication can be simulated. Concerning the former, a first approach has been developed by Klop [54], and can be called "erasure by garbage collection". It consists on accumulating unwanted data during computation in place of erasing it. For example, $\mathbf{K}' = \lambda xy.\langle x, y\rangle$ represents the classical $\mathbf{K} = \lambda xy.x$, the second component of $\langle x, y\rangle$ being garbage. Another approach is by Mackie, and can be called "erasure by data consumption" [63]. It involves a step-wise erasure process that proceeds by β-reduction until the identity $\mathbf{I}$ is eventually reached, according to the following definition:

Definition 4 (Erasability)**.** A linear λ-term M is *erasable* if $\mathcal{C}[M] \rightarrow^*_\beta \mathbf{I}$, for some context $\mathcal{C}$ such that $\mathcal{C}[M]$ is linear.

$$\texttt{let } \mathbf{I} \texttt{ be } \mathbf{I} \texttt{ in } N \to_\beta N$$
$$\texttt{let } \langle M_1, M_2\rangle \texttt{ be } x_1, x_2 \texttt{ in } N \to_\beta N[M_1/x_1, M_2/x_2]$$

(a) Reduction rules for tensors and units.

$$\frac{}{\vdash \mathbf{I} : \mathbf{1}}\ \mathbf{1}\text{R} \qquad \frac{\Gamma \vdash M : A}{\Gamma, x : \mathbf{1} \vdash \texttt{let } x \texttt{ be } \mathbf{I} \texttt{ in } M : A}\ \mathbf{1}\text{L}$$

$$\frac{\Gamma \vdash M : A \qquad \Delta \vdash N : B}{\Gamma, \Delta \vdash \langle M, N\rangle : A \otimes B}\ \otimes\text{R} \qquad \frac{\Gamma, y : A, z : B \vdash M : C}{\Gamma, x : A \otimes B \vdash \texttt{let } x \texttt{ be } y, z \texttt{ in } M : C}\ \otimes\text{L}$$

(b) Inference rules $\mathbf{1}$R, $\mathbf{1}$L, $\otimes$R and $\otimes$L for IMLL_2 in sequent style

$$\frac{}{\vdash \mathbf{I} : \mathbf{1}}\ \mathbf{1}\text{I} \qquad \frac{\Gamma \vdash N : \mathbf{1} \qquad \Delta \vdash M : A}{\Gamma, \Delta \vdash \texttt{let } N \texttt{ be } \mathbf{I} \texttt{ in } M : A}\ \mathbf{1}\text{E}$$

$$\frac{\Gamma \vdash M : A \qquad \Delta \vdash N : B}{\Gamma, \Delta \vdash \langle M, N\rangle : A \otimes B}\ \otimes\text{I} \qquad \frac{\Gamma \vdash N : A \otimes B \qquad \Gamma, y : A, z : B \vdash M : C}{\Gamma, \Delta \vdash \texttt{let } N \texttt{ be } y, z \texttt{ in } M : C}\ \otimes\text{E}$$

(c) Inference rules $\mathbf{1}$I, $\mathbf{1}$E, $\otimes$I and $\otimes$E for IMLL_2 in natural deduction style

Figure 3.2: Rules for tensors and units derivable in IMLL_2.

Example 7. The context $\mathcal{C} = (\lambda z.[\cdot])\mathbf{III}$ erases $\lambda xy.zxy$ because, filling $[\cdot]$ by $\lambda xy.zxy$, we obtain a closed linear λ-term that reduces to $\mathbf{I}$.

In [64], Mackie proves that all closed linear λ-terms can be erased by means of very simple contexts.

Lemma 4 ([64])**.** *Let $M \in \Lambda_l^\emptyset$. Then there exists $n \geq 0$ such that $M\mathbf{I}.\overset{n}{\ldots}.\mathbf{I} \to_\beta^* \mathbf{I}$.*

Proof. Since M is terminating, we assume without loss of generality that M is a normal form, and hence with shape $\lambda x_1 \ldots x_n.x_i M_1 \ldots M_m$, with $n \neq 0$ because M is closed. We proceed by induction on the size of M. If $n = 1$ and $m = 0$ then $M = \lambda x.x$ and there is nothing to prove. Otherwise, either $n > 1$ or both $n = 1$ and $m > 0$. Then we can construct the term $M\mathbf{I}.\overset{n}{\ldots}.\mathbf{I}$, and we reduce it to obtain the term $\mathbf{I}M'_1 \ldots M'_m$, where $M'_i \triangleq M_i[\mathbf{I}/x_1, \ldots, \mathbf{I}/x_n]$. It is easy to check that $|\mathbf{I}M'_1 \ldots M'_m| = |M|$. So, by reducing the head redex we get a term with strictly smaller size than M. We normalize it and we apply the induction hypothesis. □

Remark 1. We recall from [10] that a closed λ-term M is said *solvable* if, for some n, there exist λ-terms $N_1, \ldots, N_n$ such that $MN_1 \ldots N_n =_\beta \mathbf{I}$. What Lemma 4 says is that every closed linear λ-term is solvable by means of a linear context.

The notion of erasability can be addressed in a more general setting.

Definition 5 (Erasable sets)**.** Let $X \subseteq \Lambda_l$. We say that X is an *erasable set* if a linear λ-term $\mathtt{E}_X$ exists such that $\mathtt{E}_X\, M \to^*_\beta \mathbf{I}$, for all $M \in X$. We call $\mathtt{E}_X$ *eraser* of X.

The following proposition states that a finite set X of linear λ-terms is erasable if and only if all its elements are closed terms.

Proposition 5. *Let $X \subseteq \Lambda_l$:*

(1) if $\mathtt{E}_X$ is an eraser of X then $FV(\mathtt{E}_X) = \emptyset$ and $X \subseteq \Lambda_l^\emptyset$.

(2) if X is a finite subset of $\Lambda_l^\emptyset$ then it is erasable.

Proof. Concerning point (1), let X be a set of linear λ-terms and let $\mathtt{E}_X$ be an eraser of X. By definition, $\mathtt{E}_X\, M \to^*_\beta \mathbf{I}$, for all $M \in X$. Since $\mathbf{I}$ is closed, by Proposition 2.(1) both $\mathtt{E}_X$ and all elements in X must be closed terms. To prove point (2), we show that any set $X = \{M_1, \dots, M_n\} \subseteq \Lambda_l^\emptyset$ is erasable. By Lemma 4, for every $i \leq n$ there exists a $k_i \geq 0$ such that $M_i \mathbf{I} \overset{k_i}{\dots} \mathbf{I} \to^*_\beta \mathbf{I}$. It suffices to set $\mathtt{E}_X \triangleq \lambda x.x\mathbf{I} \overset{k}{\dots} \mathbf{I}$, where $k = \max_{i=1}^n k_i$. □

There exist infinite sets of closed linear λ-terms that are erasable.

Example 8. The set $X = \{\lambda z.z\mathbf{I} \overset{n}{\dots} \mathbf{I} \mid n \in \mathbb{N}\}$ is erasable. Indeed, it suffices to define $\mathtt{E}_X \triangleq \lambda y.y\mathbf{I}$. For all $n \in \mathbb{N}$, we have:

$$\mathtt{E}_X\, (\lambda z.z\mathbf{I} \overset{n}{\dots} \mathbf{I}) = (\lambda y.y\mathbf{I})(\lambda z.z\mathbf{I} \overset{n}{\dots} \mathbf{I}) \to_\beta (\lambda z.z\mathbf{I} \overset{n}{\dots} \mathbf{I})\mathbf{I} \to_\beta \mathbf{I} \overset{n+1}{\dots} \mathbf{I} \to^*_\beta \mathbf{I}.$$

However, there also exist infinite sets of closed linear λ-terms that are not erasable, so that the finiteness condition in Proposition 5.(2) is required.

Example 9. Let us denote with $L^{\lambda n}$ the linear λ-term of the form $\lambda x_n \dots x_1 x.xL$, for all $n \in \mathbb{N}$ and for all $L \in \Lambda_l$. We show that, for all $L^{\lambda n} \in \Lambda_l$ and for all contexts $\mathcal{C}$ such that $\mathcal{C}[L^{\lambda n}] \in \Lambda_l$ and $\mathcal{C}[L^{\lambda n}] \to^*_\beta \mathbf{I}$, it holds that $|\mathcal{C}[L^{\lambda n}]| \geq |L^{\lambda n}| + n$. The proof is by induction on $n \in \mathbb{N}$. The case $n = 0$ is straightforward, so let $n > 0$ and let $\mathcal{C}$ be such that $\mathcal{C}[L^{\lambda n}] \in \Lambda_l$ and $\mathcal{C}[L^{\lambda n}] \to^*_\beta \mathbf{I}$. Then, for some $\mathcal{C}' = \mathcal{C}''[[\cdot]L']$ it must be that:

$$\mathcal{C}[L^{\lambda n}] \to^*_\beta \mathcal{C}'[L^{\lambda n}] = \mathcal{C}''[L^{\lambda n}L'] \to_\beta \mathcal{C}''[\lambda x_{n-1} \dots x_1 x.x(L[L'/x_n])] \to^* \mathbf{I}$$

and, by definition, $\lambda x_{n-1} \dots x_1 x.x(L[L'/x_n]) = (L[L'/x_n])^{\lambda n-1}$. By applying the induction hypothesis, $|\mathcal{C}''[(L[L'/x_n])^{\lambda n-1}]| \geq |(L[L'/x_n])^{\lambda n-1}| + (n-1)$. Hence, by using Proposition 2.(3):

$$\begin{aligned} |L^{\lambda n}| + n &\leq |L^{\lambda n}L'| + (n-1) = |(L[L'/x_n])^{\lambda n-1}| + (n-1) + 3 \leq |\mathcal{C}''[(L[L'/x_n])^{\lambda n-1}]| + 3 \\ &= |\mathcal{C}''[L^{\lambda n}L']| = |\mathcal{C}'[L^{\lambda n}]| \leq |\mathcal{C}[L^{\lambda n}]|. \end{aligned}$$

Now, suppose that $X = \{L^{\lambda n} \in \Lambda_l^\emptyset \mid n \in \mathbb{N},\ L \in \Lambda_l\}$ is erasable, and let $\mathtt{E}_X$ be an eraser of X. We define $\mathcal{C} \triangleq (\lambda x.\mathtt{E}_X x)[\cdot]$ so that, for all $L^{\lambda n} \in X$, $\mathcal{C}[L^{\lambda n}] \to_\beta \mathtt{E}_X L^{\lambda n} \to^*_\beta \mathbf{I}$. This would imply that, for all $n \in \mathbb{N}$, $|\mathtt{E}_X L^{\lambda n}| + 3 = |(\lambda x.\mathtt{E}_X x)L^{\lambda n}| = |\mathcal{C}[L^{\lambda n}]| \geq |L^{\lambda n}| + n$, which is impossible.

In the same spirit of Definition 5, we now investigate duplicability in the linear λ-calculus.

Definition 6 (Duplicable sets)**.** Let $X \subseteq \Lambda_l$. We say that X is a *duplicable set* if a linear λ-term $\mathtt{D}_X$ exists such that $\mathtt{D}_X\, M \to^*_\beta \langle M, M \rangle$ and $FV(\mathtt{D}_X) \cap FV(M) = \emptyset$, for all $M \in X$. We call $\mathtt{D}_X$ *duplicator* of X.

Proposition 6. *Let $X \subseteq \Lambda_l$. If $\mathtt{D}_X$ is a duplicator of X then $FV(\mathtt{D}_X) = \emptyset$ and X is a finite subset of $\Lambda_{\oplus}^{\emptyset}$.*

Proof. Let X be a set of linear λ-terms, and let $\mathtt{D}_X$ be a duplicator of X. By definition, $\mathtt{D}_X\, M \rightarrow_{\beta}^{*} \langle M, M\rangle$, for all $M \in X$. Since both M and $\mathtt{D}_X$ are linear λ-terms and $FV(\mathtt{D}_X) \cap FV(M) = \emptyset$, we have that $\mathtt{D}_X\, M$ is linear, for all $M \in X$. By Proposition 2.(1), if $FV(\mathtt{D}_X) \neq \emptyset$, then every free variable in $\mathtt{D}_X$ would occur in $\langle M, M\rangle$, contradicting $FV(\mathtt{D}_X) \cap FV(M) = \emptyset$. Moreover, if there were a variable occurring free in a term $M \in X$, then it would occur twice in $\langle M, M\rangle$, contradicting Proposition 2.(2). This proves that $X \subseteq \Lambda_l^{\emptyset}$. Now, suppose that a duplicator $\mathtt{D}_X$ for an infinite set X of closed linear λ-terms exists. Since we consider terms modulo α-conversion, the set $\{M \in X \mid |M| \leq n\}$ is finite for all $n \in \mathbb{N}$, so that we can always find a $M \in X$ such that $|\mathtt{D}_X| < |M|$. Then, we would have $\mathtt{D}_X\, M \rightarrow_{\beta}^{*} \langle M, M\rangle$ with $|\langle M, M\rangle| > |\mathtt{D}_X| + |M|$. This is impossible by Proposition 2.(3). □

The above proposition states that only closed and finite sets can be duplicated. We conjecture that the converse holds as well, as long as we restrict to sets of β-normal forms that are pairwise non η-convertible. Indeed, duplication in a linear setting ultimately relies on the following linear version of the general Separation Theorem for the standard λ-calculus proved by Böhm et al. [18]:

Conjecture 7 (General separation for Λ_l)**.** *Let $X = \{M_1, \ldots, M_n\}$ be a set of β-normal closed linear λ-terms which are pairwise non η-convertible. Then, for all $N_1, \ldots, N_n$ linear λ-terms, there exists a linear λ-term F such that $F\, M_i \rightarrow_{\beta}^{*} N_i$, $\forall i \leq n$.*

Now, let $X = \{M_1, \ldots, M_n\}$ be a finite set of β-normal closed linear λ-terms which are pairwise non η-convertible. If Conjecture 7 were true, by fixing $N_i \triangleq \langle M_i, M_i\rangle$ for all $i \leq n$, there would exists a linear λ-term $\mathtt{D}_X$ such that $\mathtt{D}_X\, M_i \rightarrow_{\beta}^{*} \langle M_i, M_i\rangle$.

To sum up, Remark 1 and Conjecture 7 allow us to connect linear erasure and duplication to standard λ-calculus notions:

linear solvability $\sim$ *linear erasability*

linear separation $\sim$ *linear duplication.*

3.1.3 The typed setting

Proposition 5 and Proposition 6 say that duplicable and erasable sets contain only closed terms and, moreover, that duplicable sets must be finite. In what follows we shall study uniform copying and erasing mechanisms applied to very special kinds of sets, namely the classes of closed and normal linear λ-terms that inhabit a given type of IMLL_2.

Let us begin with a simple example based on booleans. Due to the lack of explicit weakening, the standard second-order encoding of booleans by means of the type $\forall\alpha.\alpha \multimap \alpha \multimap \alpha$ is meaningless in IMLL_2. An alternative encoding provided by Mairson and Terui [65] is the following:

$$\mathbf{B} \triangleq \forall\alpha.\alpha \multimap \alpha \multimap \alpha \otimes \alpha \qquad \mathtt{tt} \triangleq \lambda x.\lambda y.\langle x, y\rangle \qquad \mathtt{ff} \triangleq \lambda x.\lambda y.\langle y, x\rangle \tag{3.2}$$

where the "truth" $\mathtt{tt}$ and the "falsity" $\mathtt{ff}$ are the only closed and normal inhabitants of the type $\mathbf{B}$. Observe that such terms implement the "erasure by garbage collection": the first element of the pair is the "real" output, while the second one is garbage.

Starting from the terms $\mathtt{tt}$ and $\mathtt{ff}$ in (3.2), Mairson shows in [64] that IMLL is expressive enough to encode boolean functions. Mairson and Terui reformulate that encoding in IMLL_2 to prove results about the complexity of cut-elimination [65], where the advantage of working in IMLL_2 is to assign uniform types to the λ-terms representing boolean functions.

The key step to obtain the encoding of boolean functions is the existence of an *eraser* $\mathtt{E}_{\mathbf{B}}$ and a *duplicator* $\mathtt{D}_{\mathbf{B}}$ for the type $\mathbf{B}$:

$$\mathtt{E}_{\mathbf{B}} \triangleq \lambda z.\mathtt{let}\ z\mathbf{II}\ \mathtt{be}\ x,y\ \mathtt{in}\ (\mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x) : \mathbf{B} \multimap \mathbf{1} \tag{3.3}$$

$$\mathtt{D}_{\mathbf{B}} \triangleq \lambda z.\pi_1(z\langle \mathtt{tt},\mathtt{tt}\rangle\langle \mathtt{ff},\mathtt{ff}\rangle) : \mathbf{B} \multimap \mathbf{B} \otimes \mathbf{B} \tag{3.4}$$

$$\pi_1 \triangleq \lambda z.\mathtt{let}\ z\ \mathtt{be}\ x,y\ \mathtt{in}\ (\mathtt{let}\ \mathtt{E}_{\mathbf{B}}\, y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x) : (\mathbf{B} \otimes \mathbf{B}) \multimap \mathbf{B} \tag{3.5}$$

where π_1 is the linear λ-term projecting the first element of a pair. When applied to a closed and normal inhabitant of $\mathbf{B}$, i.e. either $\mathtt{tt}$ or $\mathtt{ff}$, the eraser $\mathtt{E}_{\mathbf{B}}$ consumes it until an identity is reached, while the duplicator $\mathtt{D}_{\mathbf{B}}$ produces a pair containing two copies of it:

$$\mathtt{E}_{\mathbf{B}}\,\mathtt{tt} \rightarrow^*_\beta \mathbf{I} \qquad\qquad \mathtt{D}_{\mathbf{B}}\,\mathtt{tt} \rightarrow^*_\beta \langle \mathtt{tt},\mathtt{tt}\rangle$$

$$\mathtt{E}_{\mathbf{B}}\,\mathtt{ff} \rightarrow^*_\beta \mathbf{I} \qquad\qquad \mathtt{D}_{\mathbf{B}}\,\mathtt{ff} \rightarrow^*_\beta \langle \mathtt{ff},\mathtt{ff}\rangle$$

Let us remark that erasability in IMLL_2 is subtler than the untyped case, since the type-theoretical constraints force the eraser $\mathtt{E}_{\mathbf{B}}$ to make use of something more than mere stacks of identities (see Lemma 4). Also, note that *both* the possible outcomes of duplication $\langle \mathtt{tt},\mathtt{tt}\rangle$ and $\langle \mathtt{ff},\mathtt{ff}\rangle$ are built-in components of $\mathtt{D}_{\mathbf{B}}$. In accordance with the given input, $\mathtt{D}_{\mathbf{B}}$ selects the right pair representing the result by *erasing* the unwanted one. Then, this linear mechanism of duplication works *by selection and erasure*, i.e. by a stepwise elimination of useless data until the desired result shows up.

Henceforth, closed and normal linear λ-terms will be called “values”:

Definition 7 (Values)**.** A *value* is a linear λ-term that is both (β)-normal and closed. Values are ranged over by V.

The analysis of (3.3) and (3.4) leads to the following formal notions:

Definition 8 (Duplicable and erasable types in IMLL_2)**.** Let A be a type in IMLL_2:

- we say that A is an *erasable type* if a linear λ-term $\mathtt{E}_A : A \multimap \mathbf{1}$ exists such that $\mathtt{E}_A\, V \rightarrow^*_\beta \mathbf{I}$, for every value V of A, and we call $\mathtt{E}_A$ *eraser* of A;

- we say that A is a *duplicable type* if a linear λ-term $\mathtt{D}_A : A \multimap A \otimes A$ exists such that $\mathtt{D}_A\, V \rightarrow^*_{\beta\eta} \langle V, V\rangle$, for every value V of A, and we call $\mathtt{D}_A$ *duplicator* of A.

Remark 2. Duplicators in the typed setting are defined with the help of the η-reduction, as opposed to Definition 6. We shall use η-reduction in the proof of Theorem 10, since the duplicators we will construct η-expand a value before copying it. Moreover, observe that the restriction of duplication and erasure to values, i.e. to *closed* and *normal* linear λ-terms, causes no loss of generality. On the one hand, Proposition 5.(1) and Proposition 6 imply that only closed terms can be duplicated or erased linearly. On the other hand, every type has infinitely many (closed) inhabitants, while Proposition 6 states that a duplicable set of linear λ-terms must be finite. Indeed, a necessary condition for duplicable types is that the set of values that inhabit them is finite: this is not always assured in IMLL_2 as pointed out in Example 6.

The type $\mathbf{B}$ in (3.2) is a typical example of duplicable and erasable type, whose eraser and duplicator are, respectively, the terms in (3.3) and (3.4).

In [65], Mairson and Terui generalize the mechanism of linear erasure and duplication of $\mathbf{B}$ to the class of closed Π_1 types, we shall call “ground” for ease of reference:

Definition 9 (Π_1 and Σ_1 [65])**.** The following mutually defined grammars generate Π_1 and Σ_1:

$$\begin{aligned}\Pi_1 &:= \alpha \mid \Sigma_1 \multimap \Pi_1 \mid \forall\alpha.\Pi_1\\ \Sigma_1 &:= \alpha \mid \Pi_1 \multimap \Sigma_1\end{aligned}$$

Definition 10 (Ground type)**.** A *ground type* is a *closed* Π_1 type.

We note that the universal quantifier $\forall$ occurs only positively in a Π_1 type, hence in ground types.

The type $\mathbf{B}$ in (3.2), the unit $\mathbf{1}$ and the tensor $A \otimes B$ (Definition 3) are all examples of ground types, if A and B are. In fact, following [65], tensors and units can occur *also* to the left-hand side of a linear implication "$\multimap$", even in negative positions. The reason is that we can ignore them in practice, thanks to the isomorphisms:

$$((A \otimes B) \multimap C) \multimap\!\circ\, (A \multimap B \multimap C) \qquad\qquad (\mathbf{1} \multimap C) \multimap\!\circ\, C.$$

As we shall see in the next section, the set of values that inhabit a Π_1 type is always finite (see Lemma 16). So, ground types represent *finite* data types, while the values that inhabit ground types represent their data. According to this idea, the following proposition states that all values are data of some data type:

Proposition 8. *Every closed linear λ-term M has a ground type.*

Proof. Every closed linear λ-term M is typable in IMLL by Theorem 3. Types in IMLL are quantifier-free instances of Π_1 types. Hence, M has also a Π_1 type A in IMLL_2. Let $FV(A) = \{\alpha_1, \dots, \alpha_n\}$. Since M inhabits A, it also inhabits $\overline{A} = \forall\alpha_1.\cdots.\forall\alpha_n.A$, which is a closed Π_1 type, i.e. a ground type in IMLL_2. $\square$

Ground types have an eraser:

Theorem 9 ([65])**.** *Every ground type is erasable.*

Proof. The result follows from proving the statements below by simultaneous induction:

- for every Π_1 type A with free type variables $\alpha_1, \dots, \alpha_n$ there exists a linear λ-term $\mathtt{E}_A$ such that $\vdash \mathtt{E}_A : A[\mathbf{1}/\alpha_1, \dots, \mathbf{1}/\alpha_n] \multimap \mathbf{1}$;
- for every Σ_1 type A with free type variables $\alpha_1, \dots, \alpha_n$ there exists a linear λ-term $\mathtt{H}_A$ such that $\vdash \mathtt{H}_A : A[\mathbf{1}/\alpha_1, \dots, \mathbf{1}/\alpha_n]$.

If $A = \alpha$, then $A[\mathbf{1}/\alpha] = \mathbf{1}$, and we define $\mathtt{E}_A = \mathtt{H}_A \triangleq \mathbf{I}$. If $A = B \multimap C$, then $A[\mathbf{1}/\alpha_1, \dots, \mathbf{1}/\alpha_n] = B[\mathbf{1}/\alpha_1, \dots, \mathbf{1}/\alpha_n] \multimap C[\mathbf{1}/\alpha_1, \dots, \mathbf{1}/\alpha_n]$. If $B \multimap C$ is a Π_1 type, then B is a Σ_1 type and C is a Π_1 type. By induction hypothesis, $\mathtt{H}_B$ and $\mathtt{E}_C$ exist, so that we define $\mathtt{E}_{B\multimap C} \triangleq \lambda z.\mathtt{E}_C(z\mathtt{H}_B)$. Otherwise, $B \multimap C$ is a Σ_1 type, so that B is a Π_1 type and C is a Σ_1 type. By induction hypothesis, the terms $\mathtt{E}_B$ and $\mathtt{H}_C$ exist, so that we define $\mathtt{H}_{B\multimap C} \triangleq \lambda z.\mathtt{let}\ \mathtt{E}_B\, z\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ \mathtt{H}_C$. The last case is when $A = \forall\alpha.B$, and A can only be a Π_1 type. By induction hypothesis, $\mathtt{E}_B$ exist, so that we define $\mathtt{E}_{\forall\alpha.B} \triangleq \mathtt{E}_B$. $\square$

Remark 3. In [65], Mairson and Terui actually show the existence of an eraser for all closed $e\Pi_1$ types, that properly extend the class of ground types. The $e\Pi_1$ types are generated by the following grammar:

$$e\Pi_1 := \alpha \mid e\Sigma_1 \multimap e\Pi_1 \mid \forall\alpha.e\Pi_1 \tag{3.6}$$
$$e\Sigma_1 := \alpha \mid e\Pi_1 \multimap e\Sigma_1 \mid \forall\alpha.e\Sigma_1 \tag{3.7}$$

where, in the last clause of (3.7), $\forall\alpha.e\Sigma_1$ must be inhabited. Roughly, a $e\Pi_1$ type (resp. $e\Sigma_1$ type) is like a Π_1 type (resp. Σ_1 type), but it may additionally contain negative (resp. positive) occurrences of *inhabited* $\forall$ types. A typical example of closed $e\Pi_1$ type is $(\mathbf{B} \multimap \mathbf{B}) \multimap \mathbf{B}$. If the ground types are finite data types, the closed $e\Pi_1$ types can be seen as functionals over finite data types. We decided not to consider this class to give a uniform account of the mechanisms of erasability and duplication. Moreover, several remarkable closed $e\Pi_1$ types, like for example $(\mathbf{B} \multimap \mathbf{B}) \multimap \mathbf{B}$, are built up from (inhabited) ground types and linear implications, so that their erasers can be easily built from erasers of these latter. As an example, an eraser of $(\mathbf{B} \multimap \mathbf{B}) \multimap \mathbf{B}$ is $\lambda x.\mathtt{E}_{\mathbf{B}}(x\mathbf{I})$.

Inhabited ground types have a duplicator:

Theorem 10. *Every inhabited ground type is duplicable.*

Mairson and Terui sketch the proof of Theorem 10 in [65]. In the next section we shall develop it in every detail.

3.2 The Duplication Theorem

In this section we give a detailed proof of Theorem 10 for IMLL_2, which states that if A is an inhabited ground type, i.e an inhabited closed Π_1 type, then A is also a duplicable type. By Definition 8, this amounts to show that a linear λ-term $\mathtt{D}_A : A \multimap A \otimes A$ exists such that $\mathtt{D}_A\, V \rightarrow^*_{\beta\eta} \langle V, V\rangle$ holds for every value V of A.

We shall construct the duplicator $\mathtt{D}_A$ of a ground type A as the composition of three linear λ-terms, diagrammatically displayed in Figure 3.3. Taking a value V of type A as input, $\mathtt{D}_A$ implements the following three main operations:

(1) expand V to its η-long normal form V_A;

(2) compile V_A to a linear λ-term $\lceil V_A \rceil$ which encodes V_A as a boolean tuple;

(3) copy and decode $\lceil V_A \rceil$, obtaining the duplication $\langle V_A, V_A\rangle$ of V_A, that η-reduces to $\langle V, V\rangle$.

Point (3) is the one implementing Mairson and Terui's "duplication by selection and erasure" discussed in the previous section. In particular, duplication is by means of the term $\mathtt{dec}^s_A$ in Section 3.2.3. It nests a series of `if-then-else` constructs which is a look-up table, possibly quite big, that stores all the pairs of normal inhabitants of A. Each of them represents a possible outcome of the duplication. Given a boolean tuple $\lceil V_A \rceil$ in input, the nested `if-then-else` select the corresponding pair $\langle V_A, V_A\rangle$, erasing all the remaining "candidates". The inhabitation condition for A stated in Theorem 10 ensures that the default pair $\langle V', V'\rangle$ exists as a sort of "exception". We "throw" it each time the boolean tuple that $\mathtt{dec}^s_A$ receives as input does not encode any value of type A.

For each such component of $\mathtt{D}_A$ we dedicate a specific subsection. For the sake of presentation, in this section we focus on terms of IMLL_2 rather than on derivations, so that when we say that a term M has type A with context Γ, we clearly mean that a derivation $\mathcal{D}$ exists such that $\mathcal{D} \lhd \Gamma \vdash M : A$. Moreover, as assumed in Section 2.5.3, terms are considered modulo α-equivalence.

3.2.1 The linear λ-term $\mathtt{sub}^s_A$

Roughly, the λ-term $\mathtt{sub}^s_A$, when applied to a value V of ground type A, produces its η-long normal form V_A whose type is obtained from A as follows: we strip away every occurrence of $\forall$

$$\begin{tikzcd}
A \arrow[r, "\mathtt{sub}^s_A"] & A^-[\mathbf{B}^s] \arrow[r, "\mathtt{enc}^s_A"] & \mathbf{B}^s \arrow[r, "\mathtt{dec}^s_A"] & A \otimes A \\
V \arrow[r, mapsto] & V_A \arrow[r, mapsto] & \lceil V_A \rceil \arrow[r, mapsto] & \langle V_A, V_A \rangle
\end{tikzcd}$$

Figure 3.3: The diagrammatic representation of $\mathtt{D}_A$.

and we substitute each type variable with the s-ary tensor of boolean datatypes $\mathbf{B}^s = \mathbf{B}\otimes.\overset{s}{.}.\otimes\mathbf{B}$, for some $s > 0$.

Before introducing the λ-term $\mathtt{sub}^s_A$, we need the definition of η-long normal form:

Definition 11 (η-long normal forms)**.** Let $\mathcal{D} \lhd \Gamma \vdash M : B$ be cut-free. We define the *η-expansion* of $\mathcal{D}$, denoted $\mathcal{D}^\Gamma_B$, as the derivation obtained from $\mathcal{D}$ by substituting every occurrence of:

$$\frac{}{x : A \vdash x : A}\ ax$$

with a derivation of $x : A \vdash M' : A$, for some M', whose axioms have form $y : \alpha \vdash y : \alpha$. Given $\mathcal{D} \lhd \Gamma \vdash M : B$, its η-expansion $\mathcal{D}^\Gamma_B$ is unique, and its concluding judgement is denoted by $\Gamma \vdash M^\Gamma_B : B$. The term M^Γ_B is called *η-long normal form* and is such that $M^\Gamma_B \rightarrow^*_\eta M$. If the context Γ of an η-expanded $\mathcal{D}$ is $x_1 : A_1, \dots, x_n : A$ we may write $\mathcal{D}^{A_1,\dots,A_n}_B$ and $M^{A_1,\dots,A_n}_B$. If Γ is empty, we feel free to write $\mathcal{D}_B$ and M_B.

Lemma 11. *Let $\mathcal{D} \lhd \Gamma \vdash M : A$ be a cut-free derivation in IMLL_2, and let M^Γ_A denote the η-long normal form obtained by η-expanding $\mathcal{D}$. Then:*

(1) If $M = x$, $A = \alpha$, and $\Gamma = x : \alpha$ then $x^\alpha_\alpha = x$.

(2) If $M = x$, $A = \forall\alpha.B$, and $\Gamma = x : \forall\alpha.B$ then $x^{\forall\alpha.B}_{\forall\alpha.B} = x^B_B$.

(3) If $M = x$, $A = B \multimap C$, and $\Gamma = x : B \multimap C$ then $x^{B\multimap C}_{B\multimap C} = \lambda y.(xy^B_B)^C_C$.

(4) If $A = \forall\alpha.B$ then $M^\Gamma_{\forall\alpha.B} = M^\Gamma_{B\langle\gamma/\alpha\rangle}$, for some fresh type variable γ.

(5) If $M = \lambda x.N$ and $A = B \multimap C$ then $(\lambda x.N)^\Gamma_{B\multimap C} = \lambda x.N^{\Gamma,x:B}_C$.

(6) If $M = P[yN/x]$ and $\Gamma = \Delta, \Sigma, y : B \multimap C$, where P has type A with context $\Delta, x : C$ and N has type B with context Σ, then $(P[yN/x])^\Gamma_A = P^{\Delta,x:C}_A[yN^\Sigma_B/x]$.

(7) If $M = P[yN/x]$ and $\Gamma = \Gamma', y : \forall\alpha.B$ then $(P[yN/x])^{\Gamma',y:\forall\alpha.B}_A = (P[yN/x])^{\Gamma',y:B\langle D/\alpha\rangle}_A$, for some suitable type D.

Proof. Just follow the definition of η-long normal form. □

Definition 12 (The map $(\_)^-$)**.** Let A be a type in IMLL_2. We define A^- by induction on the complexity of the type:

$$\alpha^- \triangleq \alpha$$
$$(A \multimap B)^- \triangleq A^- \multimap B^-$$
$$(\forall\alpha.A)^- \triangleq A^-\langle\gamma/\alpha\rangle$$

where γ is taken from the head of an infinite list of fresh type variables. The notation $A[B]$ denotes the type obtained by replacing B for every free type variable of A. Moreover, if $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, then Γ^- stands for $x_1 : A_1^-, \ldots, x_n : A_n^-$, and $\Gamma[B]$ stands for $x_1 : A_1[B], \ldots, x_n : A_n[B]$.

Lemma 12. *Let A be a Π_1 type and $A_1, \ldots, A_n$ be Σ_1 types. If $x_1 : A_1, \ldots, x_n : A_n \vdash M : A$ then, for every type B, $x_1 : A_1^-[B], \ldots, x_n : A_n^-[B] \vdash M : A^-[B]$.*

Proof. Easy induction on a derivation of $x_1 : A_1, \ldots, x_n : A_n \vdash M : A$. $\square$

Definition 13 (The linear λ-term $\mathtt{sub}^s_A$)**.** Let $s > 0$. We define the linear λ-terms $\mathtt{sub}^s_A : A[\mathbf{B}^s] \multimap A^-[\mathbf{B}^s]$, where A is a Π_1 type, and $\overline{\mathtt{sub}}^s_A : A^-[\mathbf{B}^s] \multimap A[\mathbf{B}^s]$, where A is a Σ_1 type, by simultaneous induction on the size of A:

$$\mathtt{sub}^s_\alpha \triangleq \lambda x.x \qquad\qquad \overline{\mathtt{sub}}^s_\alpha \triangleq \lambda x.x$$

$$\mathtt{sub}^s_{\forall\alpha.B} \triangleq \mathtt{sub}^s_B$$

$$\mathtt{sub}^s_{B\multimap C} \triangleq \lambda x.\lambda y.\mathtt{sub}^s_C(x\,(\overline{\mathtt{sub}}^s_B\, y)) \qquad\qquad \overline{\mathtt{sub}}^s_{B\multimap C} \triangleq \lambda x.\lambda y.\overline{\mathtt{sub}}^s_C(x\,(\mathtt{sub}^s_B\, y)).$$

The following will be used to compact the proof of some of the coming lemmas.

Definition 14. Let $s > 0$. Let A be a Π_1 type, and let $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ be a context of Σ_1 types. If M is an inhabitant of $A[\mathbf{B}^s]$ with context $\Gamma[\mathbf{B}^s]$, then $M[\Gamma]$ denotes the substitution:

$$M[\overline{\mathtt{sub}}^s_{A_1}\, x_1'/x_1, \ldots, \overline{\mathtt{sub}}^s_{A_n}\, x_n'/x_n]$$

for some $x_1', \ldots, x_n'$.

Lemma 13. *Let $s > 0$ and z be of type $A[\mathbf{B}^s]$.*

(1) If A is a Π_1 type, then $\mathtt{sub}^s_A\, z \to^*_\beta z^A_A$.

(2) If A is a Σ_1 type, then $\overline{\mathtt{sub}}^s_A\, z \to^*_\beta z^A_A$.

Proof. We prove both points by simultaneous induction on $|A|$:

- Case $A = \alpha$. Both the statements are straightforward since we have $z^\alpha_\alpha = z$ by Lemma 11.(1).
- Case $A = \forall\alpha.B$. This case applies to point (1) only. By induction hypothesis, for every variable x of type $B[\mathbf{B}^s]$, $\mathtt{sub}^s_B\, x \to^*_\beta x^B_B$. The λ-term $\mathtt{sub}^s_B$ has type $B[\mathbf{B}^s] \multimap B^-[\mathbf{B}^s]$, which is equal to $(B\langle\mathbf{B}^s/\alpha\rangle)[\mathbf{B}^s] \multimap (\forall\alpha.B)^-[\mathbf{B}^s]$. Hence, $\mathtt{sub}^s_B$ has also type $(\forall\alpha.B)[\mathbf{B}^s] \multimap (\forall\alpha.B)^-[\mathbf{B}^s]$. Moreover, by Definition 13 we have $\mathtt{sub}^s_B = \mathtt{sub}^s_{\forall\alpha.B}$. Therefore, for every variable z of type $(\forall\alpha.B)[\mathbf{B}^s]$ we have $\mathtt{sub}^s_{\forall\alpha.B}\, z = \mathtt{sub}^s_B\, z \to^*_\beta z^B_B$. But $z^B_B = z^{\forall\alpha.B}_{\forall\alpha.B}$ by Lemma 11.(2).
- Case $A = B \multimap C$. We prove point (1) only (point (2) is similar). Let z be of type $(B \multimap C)[\mathbf{B}^s] = B[\mathbf{B}^s] \multimap C[\mathbf{B}^s]$. Then we have

$$\begin{aligned}
\mathtt{sub}^s_{B\multimap C}\, z &= (\lambda x.\lambda y.\mathtt{sub}^s_C(x(\overline{\mathtt{sub}}^s_B\, y)))z && \text{Def. 13}\\
&\to_\beta \lambda y.\mathtt{sub}^s_C(z(\overline{\mathtt{sub}}^s_B\, y))\\
&= \lambda y.(\mathtt{sub}^s_C\, w)[z(\overline{\mathtt{sub}}^s_B\, y)/w]\\
&\to^*_\beta \lambda y.w^C_C[z(\overline{\mathtt{sub}}^s_B\, y)/w] && \text{IH on point (1)}\\
&\to^*_\beta \lambda y.w^C_C[zy^B_B/w] && \text{IH on point (2)}\\
&= \lambda y.(zy^B_B)^C_C\\
&= z^{B\multimap C}_{B\multimap C}. && \text{Lem. 11.(3)}
\end{aligned}$$

□

Lemma 14. *Let $s > 0$. If $z : A[\mathbf{B}^s]$, where A is a Π_1 type, then $\mathtt{sub}^s_A\, z^A_A \to^*_\beta z^A_A$.*

Proof. We prove it by induction on $|A|$:

- Case $A = \alpha$. The statement is straightforward since we have $z^\alpha_\alpha = z$ by Lemma 11.(1)
- Case $A = \forall\alpha.B$. By Definition 13, $\mathtt{sub}^s_{\forall\alpha.B} = \mathtt{sub}^s_B$ and we use the induction hypothesis.
- Case $A = B \multimap C$. Then we have

$$\begin{aligned}
\mathtt{sub}^s_{B\multimap C}\, z^{B\multimap C}_{B\multimap C} &= (\lambda x.\lambda y.\mathtt{sub}^s_C(x(\overline{\mathtt{sub}}^s_B\, y)))z^{B\multimap C}_{B\multimap C} && \text{Def. 13}\\
&= (\lambda x.\lambda y.\mathtt{sub}^s_C(x(\overline{\mathtt{sub}}^s_B\, y)))(\lambda w.(zw^B_B)^C_C) && \text{Lem. 11.(3)}\\
&\to_\beta \lambda y.\mathtt{sub}^s_C((\lambda w.(zw^B_B)^C_C)(\overline{\mathtt{sub}}^s_B\, y))\\
&\to_\beta \lambda y.\mathtt{sub}^s_C(z(\overline{\mathtt{sub}}^s_B\, y)^B_B)^C_C\\
&\to^*_\beta \lambda y.\mathtt{sub}^s_C(zy^B_B)^C_C && \text{Lem. 13.(2)}\\
&= \lambda y.(\mathtt{sub}^s_C\, w^C_C)[zy^B_B/w]\\
&\to^*_\beta \lambda y.w^C_C[zy^B_B/w] && \text{IH}\\
&= \lambda y.(zy^B_B)^C_C\\
&=_\alpha z^{B\multimap C}_{B\multimap C}. && \text{Lem. 11.(3)}
\end{aligned}$$

□

Lemma 15. *Let $s > 0$. Let A be a Π_1 type, and let $\Gamma = x_1 : A_1, \dots, x_n : A_n$ be a context of Σ_1 types. If $\Gamma[\mathbf{B}^s] \vdash M : A[\mathbf{B}^s]$, with M normal, then:*

$$\mathtt{sub}^s_A\, M[\Gamma] \to^*_\beta M^\Gamma_A.$$

Proof. Let $Q_{\Gamma,A}$ be the number of universal quantifications in $A_1, \dots, A_n, A$. We prove the result by induction on $|M| + Q_{\Gamma,A}$. If $M = z$ then $\Gamma = z : A$ and $\mathtt{sub}^s_A\, M[\Gamma] = \mathtt{sub}^s_A(\overline{\mathtt{sub}}^s_A\, z)$. By point (2) of Lemma 13 and by Lemma 14 we have $\mathtt{sub}^s_A(\overline{\mathtt{sub}}^s_A\, z) \to^*_\beta \mathtt{sub}^s_A\, z^A_A \to^*_\beta z^A_A$. If $M = \lambda z.N$ then we have two cases depending on the type of M:

- Case $A = \forall\alpha.B$. The λ-term $\mathtt{sub}^s_B$ has type $B[\mathbf{B}^s] \multimap B^-[\mathbf{B}^s]$, i.e. $(B\langle\mathbf{B}^s/\alpha\rangle)[\mathbf{B}^s] \multimap (\forall\alpha.B)^-[\mathbf{B}^s]$, so that $\mathtt{sub}^s_B$ has also type $(\forall\alpha.B)[\mathbf{B}^s] \multimap (\forall\alpha.B)^-[\mathbf{B}^s]$. By Definition 13 we have $\mathtt{sub}^s_B = \mathtt{sub}^s_{\forall\alpha.B}$. By using the induction hypothesis, for every M of type $(\forall\alpha.B)[\mathbf{B}^s]$ with context $\Gamma[\mathbf{B}^s]$, we have $\mathtt{sub}^s_{\forall\alpha.B}\, M[\Gamma] = \mathtt{sub}^s_B\, M[\Gamma] \to^*_\beta M^\Gamma_B$. Moreover, by Lemma 11.(4), $M^\Gamma_B = M^\Gamma_{\forall\alpha.A}$.
- Case $A = B \multimap C$. Then we have:

$$\begin{aligned}
\mathtt{sub}^s_{B\multimap C}\, M[\Gamma] &= (\lambda x.\lambda y.\mathtt{sub}^s_C(x\,(\overline{\mathtt{sub}}^s_B\, y)))(\lambda z.N)[\Gamma] && \text{Def. 13}\\
&\to_\beta \lambda y.\mathtt{sub}^s_C((\lambda z.N)[\Gamma]\,(\overline{\mathtt{sub}}^s_B\, y))\\
&\to_\beta \lambda y.\mathtt{sub}^s_C((N[\Gamma])[\overline{\mathtt{sub}}^s_B\, y/z])\\
&= \lambda y.\mathtt{sub}^s_C(N[\Gamma, y : B]) && \text{Def. 14}\\
&\to^*_\beta \lambda y.N^{\Gamma,y:B}_C && \text{IH}\\
&= (\lambda y.N)^\Gamma_{B\multimap C} && \text{Lem. 11.(5)}\\
&=_\alpha M^\Gamma_{B\multimap C}.
\end{aligned}$$

If $M = P[zN/w]$ then the type of z cannot have an outermost universal quantification, because Γ is a context of Σ_1 types. So z has type of the form $B \multimap C$ in Γ. Let Γ' and Γ'' be contexts such that $\Gamma = \Gamma', \Gamma'', z : B \multimap C$, $\mathrm{dom}(\Gamma') = FV(P)$, and $\mathrm{dom}(\Gamma'') = FV(N)$. Then we have:

$$\begin{aligned}
\mathtt{sub}^s_A\, M[\Gamma] &= \mathtt{sub}^s_A(P[zN/w])[\Gamma] \\
&= \mathtt{sub}^s_A(P[\Gamma'][(zN)[\Gamma'', z : B \multimap C]/w]) \\
&= \mathtt{sub}^s_A(P[\Gamma'][(\overline{\mathtt{sub}}^s_{B\multimap C}\, z)(N[\Gamma''])/w]) \\
&\to^*_\beta \mathtt{sub}^s_A(P[\Gamma'][\overline{\mathtt{sub}}^s_C(z\,(\mathtt{sub}^s_B\,(N[\Gamma''])))/w]) && \text{Def. 13} \\
&\to^*_\beta \mathtt{sub}^s_A(P[\Gamma'][\overline{\mathtt{sub}}^s_C(zN^{\Gamma''}_B)/w]) && \text{IH} \\
&= (\mathtt{sub}^s_A\, P[\Gamma'][\overline{\mathtt{sub}}^s_C\, w/w])[zN^{\Gamma''}_B/w] \\
&= (\mathtt{sub}^s_A\, P[\Gamma', w : C])[zN^{\Gamma''}_B/w] && \text{Def. 14} \\
&\to^*_\beta P^{\Gamma', w:C}_A[zN^{\Gamma''}_B/w] && \text{IH} \\
&= (P[zN/w])^\Gamma_A. && \text{Lem. 11.(6)}
\end{aligned}$$

□

3.2.2 The linear λ-term $\mathtt{enc}^s_A$

A missing ingredient in the previous subsection is the value of s, which is fixed to some strictly positive integer. To determine s we need the following property:

Lemma 16 (Types bound terms)**.** *For every cut-free derivation $\mathcal{D} \lhd \Gamma \vdash M : B$ in IMLL_2 which does not contain applications of $\forall$L, the following inequations hold:*

$$|M| \leq |M^\Gamma_B| \leq |\Gamma^-| + |B^-| \leq 2 \cdot |M^\Gamma_B| \tag{3.8}$$

where $(\_)^-$ is as in Definition 12, and M^Γ_A is as in Definition 11.

Proof. The inequation $|M| \leq |M^\Gamma_B|$ is by definition of η-long normal form. Now, let $\mathcal{D}^\Gamma_B$ be the η-expansion of $\mathcal{D}$, so that $\mathcal{D}^\Gamma_B \lhd \Gamma \vdash M^\Gamma_B : B$. We prove the remaining two inequations by induction on $\mathcal{D}^\Gamma_B$. If it is an axiom then, by definition of η-expansion, it must be of the form $x : \alpha \vdash x : \alpha$, where $M^\Gamma_B = x$. Hence, $|x| \leq 2 \cdot |\alpha| \leq 2 \cdot |x|$. Both the rules $\multimap$R and $\multimap$L, increase by one the overall size of the types in a judgment and of the corresponding term, so the inequalities still hold. Last, the rules for $\forall$ do not affect the size of both Γ^-, B^- and M^Γ_B. □

Notice that Lemma 16 does not hold in general if $\mathcal{D}$ contains instances of the inference rule $\forall$L, since one can exploit the inference rule $\forall$L to "compress" the size of a type.

Now, consider a cut-free derivation $\mathcal{D} \lhd\ \vdash M : A$, where A is a ground type. Since negative occurrences of $\forall$ are not allowed in A, $\mathcal{D}$ contains no application of $\forall$L and, by Lemma 16, this implies that $|M| \leq |A^-|$. This limits the number of variables a generic inhabitant of A has, so that we can safely say that the variables of M must certainly belong to a fixed set $\{\mathrm{x}_1, \ldots, \mathrm{x}_{|A^-|}\}$. The next step is to show that we can encode every normal form as a tuple of booleans, i.e. as elements in $\mathbf{B}^s$ with a sufficiently large s. Actually, we are interested in η-long normal forms only, due to the way the linear λ-term $\mathtt{sub}^s_A$ acts on inhabitants of A as shown in the previous subsection. So, given a ground type A, we can represent the η-long normal forms of type A with tuples of type $\mathbf{B}^{\mathcal{O}(|A^-| \cdot \log|A^-|)}$, since each such linear λ-term has at most $|A^-|$ symbols, each one encoded using around $\log|A^-|$ bits. By setting $s = c \cdot (|A^-| \cdot \log|A^-|)$ for some $c > 0$ large enough, there must exist a coding function $\ulcorner\_\urcorner : \Lambda_s \longrightarrow \mathbf{B}^s$, where Λ_s is the set of all normal linear λ-terms having size bounded by s. The role of the λ-term $\mathtt{enc}^s_A$ is to internalize the coding function $\lceil\_\rceil$ in IMLL_2 as far as the η-long normal forms of a fixed type A are concerned.

The coming Lemma 18 relies on an iterated selection mechanism, i.e. a nested `if-then-else` construction. In order to define selection, we first we need to extend the projection in (3.5).

Definition 15 (Generalized projection)**.** Let A be a ground type. For all $k \geq 0$ and $\vec{m} = m_1, \dots, m_k \geq 0$, the linear λ-term $\pi_1^{\vec{m}}$ is defined below:

$$\pi_1^{\vec{m}} \triangleq \begin{cases} \lambda z.\mathtt{let}\ z\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ \mathrm{E}_A\, y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x) & \text{if } k = 0 \\ \lambda z.\mathtt{let}\ z\ \mathtt{be}\ x, y\ \mathtt{in}\ (\mathtt{let}\ \mathrm{E}_A\, (y\, \mathtt{tt}^{m_1} \dots \mathtt{tt}^{m_k})\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x) & \text{if } k > 0 \end{cases}$$

with type $B \otimes B \multimap B$, where $B \triangleq \mathbf{B}^{m_1} \multimap \dots \multimap \mathbf{B}^{m_k} \multimap A$. When $k = 0$ we simply write π_1 in place of $\pi_1^{\vec{m}}$, whose type is $A \otimes A \multimap A$.

Definition 16 (Generalized selection)**.** Let A be a ground type and let $M_{\mathtt{tt}^n}$, $M_{\langle \mathtt{tt}^{n-1}, \mathtt{ff}\rangle}$, $\dots$, $M_{\langle \mathtt{tt}, \mathtt{ff}^{n-1}\rangle}$, $M_{\mathtt{ff}^n}$ be (not necessarily distinct) normal inhabitants of $\mathbf{B}^{m_1} \multimap \dots \multimap \mathbf{B}^{m_k} \multimap A$, for some $n \geq 1$, $k \geq 0$, and $\vec{m} = m_1, \dots, m_k \geq 0$. We define the linear λ-term:

$$\mathtt{if}\ x\ \mathtt{then}\ [M_{\mathtt{tt}^n}, M_{\langle \mathtt{tt}^{n-1}, \mathtt{ff}\rangle}, \dots, M_{\langle \mathtt{tt}, \mathtt{ff}^{n-1}\rangle}, M_{\mathtt{ff}^n}]^{\vec{m}} \tag{3.9}$$

with type $\mathbf{B}^{m_1} \multimap \dots \multimap \mathbf{B}^{m_k} \multimap A$ and context $x : \mathbf{B}^n$ by induction on n:

- $n = 1$: $\mathtt{if}\ x\ \mathtt{then}\ [M_{\mathtt{tt}}, M_{\mathtt{ff}}]^{\vec{m}} \triangleq \pi_1^{\vec{m}}(x\, M_{\mathtt{tt}}\, M_{\mathtt{ff}})$.
- $n > 1$: $\mathtt{if}\ x\ \mathtt{then}\ [M_{\mathtt{tt}^n}, M_{\langle \mathtt{tt}^{n-1}, \mathtt{ff}\rangle}, \dots, M_{\langle \mathtt{tt}, \mathtt{ff}^{n-1}\rangle}, M_{\mathtt{ff}^n}]^{\vec{m}} \triangleq$

$$\begin{aligned} &\mathtt{let}\ x\ \mathtt{be}\ x_1, x_2\ \mathtt{in}\ (\mathtt{if}\ x_2\ \mathtt{then} \\ &\big[(\lambda y_1.\mathtt{if}\ y_1\ \mathtt{then}\ [P_{\mathtt{tt}^{n-1}}, P_{\langle \mathtt{tt}^{n-2}, \mathtt{ff}\rangle}, \dots, P_{\langle \mathtt{tt}, \mathtt{ff}^{n-2}\rangle}, P_{\mathtt{ff}^{n-1}}]^{\vec{m}}), \\ &\ (\lambda y_2.\mathtt{if}\ y_2\ \mathtt{then}\ [Q_{\mathtt{tt}^{n-1}}, Q_{\langle \mathtt{tt}^{n-2}, \mathtt{ff}\rangle}, \dots, Q_{\langle \mathtt{tt}, \mathtt{ff}^{n-2}\rangle}, Q_{\mathtt{ff}^{n-1}}]^{\vec{m}})\big]^{n-1, \vec{m}})\, x_1 \end{aligned}$$

  where, $\pi_1^{\vec{m}}$ is as in Definition 15 and, for every n-tuple $\langle \mathtt{b}_1, \dots, \mathtt{b}_n\rangle$ of booleans, $P_{\langle \mathtt{b}_1, \dots, \mathtt{b}_n\rangle} \triangleq M_{\langle\langle \mathtt{b}_1, \dots, \mathtt{b}_n\rangle, \mathtt{tt}\rangle}$, $Q_{\langle \mathtt{b}_1, \dots, \mathtt{b}_n\rangle} \triangleq M_{\langle\langle \mathtt{b}_1, \dots, \mathtt{b}_n\rangle, \mathtt{ff}\rangle}$.

when $k = 0$ we feel free of ruling out the apex $\vec{m}$ in (3.9).

Lemma 17. *Let A be a ground type and let $M_{\mathtt{tt}^n}$, $M_{\langle \mathtt{tt}^{n-1}, \mathtt{ff}\rangle}$, $\dots$, $M_{\langle \mathtt{tt}, \mathtt{ff}^{n-1}\rangle}$, $M_{\mathtt{ff}^n}$ be (not necessarily distinct) normal inhabitants of $\mathbf{B}^{m_1} \multimap \dots \multimap \mathbf{B}^{m_k} \multimap A$, for some $n \geq 1$, $k \geq 0$, and $\vec{m} = m_1, \dots, m_k \geq 0$. For every n-tuple of booleans $\langle \mathtt{b}_1, \dots, \mathtt{b}_n\rangle$ it holds that:*

$$\mathtt{if}\ \langle \mathtt{b}_1, \dots, \mathtt{b}_n\rangle\ \mathtt{then}\ (M_{\mathtt{tt}^n}, M_{\langle \mathtt{tt}^{n-1}, \mathtt{ff}\rangle}, \dots, M_{\langle \mathtt{tt}, \mathtt{ff}^{n-1}\rangle}, M_{\mathtt{ff}^n}) \rightarrow^*_\beta M_{\langle \mathtt{b}_1, \dots, \mathtt{b}_n\rangle}.$$

Proof. Straightforward. □

Notice that, if $n = 1$ and $k = 0$ in Definition 16, we get the usual `if-then-else` construction defined in [40] as:

$$\mathtt{if}\ x\ \mathtt{then}\ M_1\ \mathtt{else}\ M_2 \triangleq \pi_1(x\, M_1\, M_2) \tag{3.10}$$

with type A and context $x : \mathbf{B}$, where $\pi_1 : A \otimes A \multimap A$ is as in Definition 15. Clearly, if $\mathtt{b}_1 \triangleq \mathtt{tt}$ and $\mathtt{b}_2 \triangleq \mathtt{ff}$, then $\mathtt{if}\ \mathtt{b}_i\ \mathtt{then}\ M_1\ \mathtt{else}\ M_2 \rightarrow^*_\beta M_i$ for $i = 1, 2$.

Before defining the linear λ-term $\mathtt{enc}^s_A$ we need to encode the λ-abstractions and the applications in IMLL_2.

Lemma 18. *Let $s > 0$. The following statements hold:*

(1) A linear λ-term $\mathtt{abs}^s : \mathbf{B}^s \multimap \mathbf{B}^s \multimap \mathbf{B}^s$ exists such that $\mathtt{abs}\lceil x\rceil\lceil M\rceil \to_\beta^* \lceil \lambda x.M\rceil$, if $|\lambda x.M| \leq s$ and $x \in \{\mathrm{x}_1, \dots, \mathrm{x}_s\}$.

(2) A linear λ-term $\mathtt{app}^s : \mathbf{B}^s \multimap \mathbf{B}^s \multimap \mathbf{B}^s$ exists such that $\mathtt{app}\lceil M\rceil\lceil N\rceil \to_\beta^* \lceil MN\rceil$, if $|MN| \leq s$.

Proof. We sketch the proof of point (1) only, since point (2) is similar. We let $\mathtt{b}$ denote the encoding of the boolean value b in IMLL_2. The linear λ-term $\mathtt{abs}$ is of the form:

$$\lambda x.\lambda y.(\mathtt{if}\ x\ \mathtt{then}\ [P_{\mathtt{tt}^s}, P_{\langle \mathtt{tt}^{s-1}, \mathtt{ff}\rangle}, \dots, P_{\langle \mathtt{ff}, \mathtt{tt}^{s-1}\rangle}, P_{\mathtt{ff}^s}]^s)\, y$$

where, for all s-tuple of booleans $T = \langle \mathtt{b}_1, \dots, \mathtt{b}_s\rangle$, the linear λ-term P_T with type $\mathbf{B}^s \multimap \mathbf{B}^s$ is as follows:

$$\lambda y.\mathtt{if}\ y\ \mathtt{then}\ [Q^T_{\mathtt{tt}^s}, Q^T_{\langle \mathtt{tt}^{s-1}, \mathtt{ff}\rangle}, \dots, Q^T_{\langle \mathtt{ff}, \mathtt{tt}^{s-1}\rangle}, Q^T_{\mathtt{ff}^s}]$$

For all $T = \langle \mathtt{b}_1, \dots, \mathtt{b}_s\rangle$ and for all $T' = \langle \mathtt{b}'_1, \dots, \mathtt{b}'_s\rangle$ we define:

$$Q^T_{T'} = \begin{cases} \lceil \lambda x.M\rceil & \text{if } \langle \mathtt{b}_1, \dots, \mathtt{b}_s\rangle = \lceil x\rceil,\ \langle \mathtt{b}'_1, \dots, \mathtt{b}'_s\rangle = \lceil M\rceil, \\ & \text{and } |\lambda x.M| \leq s \\ & \\ \langle \mathtt{tt}, \overset{s}{\dots}, \mathtt{tt}\rangle & \text{otherwise.} \end{cases}$$

□

The λ-term $\mathtt{enc}^s_A$, given a value V_A in η-long normal form and of type A, combines the λ-terms $\mathtt{abs}^s$ and $\mathtt{app}^s$ to construct its encoding.

Definition 17 (The linear λ-term $\mathtt{enc}^s_A$)**.** Let $s > 0$. We define the linear λ-terms $\mathtt{enc}^s_A : A^-[\mathbf{B}^s] \multimap \mathbf{B}^s$, where A is a Π_1 type, and $\overline{\mathtt{enc}}^s_A : \mathbf{B}^s \multimap A^-[\mathbf{B}^s]$, where A is a Σ_1 type, by simultaneous induction on the size of A:

$$\begin{aligned} \mathtt{enc}^s_\alpha &\triangleq \lambda z.z & \mathtt{enc}^s_{B\multimap C} &\triangleq \lambda z.\mathtt{abs}^s\lceil x\rceil\,(\mathtt{enc}^s_C\,(z\,(\overline{\mathtt{enc}}^s_B\,\lceil x\rceil))) \\ \overline{\mathtt{enc}}^s_\alpha &\triangleq \lambda z.z & \overline{\mathtt{enc}}^s_{B\multimap C} &\triangleq \lambda z.\lambda x.\overline{\mathtt{enc}}^s_C\,(\mathtt{app}^s z\,(\mathtt{enc}^s_B\, x)) \end{aligned}$$

with x chosen fresh in $\{\mathrm{x}_1, \dots, \mathrm{x}_s\}$.

The following will be used to compact the proof of some of the coming lemmas.

Definition 18. Let $s > 0$, and let A be a Π_1 type and $\Gamma = x_1 : A_1, \dots, x_n : A_n$ be a context of Σ_1 types. If M is an inhabitant of type $A^-[\mathbf{B}^s]$ with context $\Gamma^-[\mathbf{B}^s]$ then $M[\Gamma]$ denotes the substitution:

$$M[\overline{\mathtt{enc}}^s_{A_1}\, x'_1/x_1, \dots, \overline{\mathtt{enc}}^s_{A_n}\, x'_n/x_n]$$

for some $x'_1, \dots, x'_n$.

To prove that $\mathtt{enc}^s_A$ is able to encode a value V_A of type A we need an intermediate step. We first prove that $\mathtt{enc}^s_A$ substitutes every λ-abstraction in V_A with an instance of $\mathtt{abs}^s$, and every application with an instance of $\mathtt{app}^s$, thus producing a "precode". Then we prove that, when every free variable in it has been substituted with its respective encoding, the precode reduces to $\lceil V_A\rceil$.

Definition 19. Let $s > 0$. If M is a linear λ-term in normal form such that $|M| \leq s$, we define M^s by induction on $|M|$:

- $M = x$ if and only if $M^s = x$,
- $M = \lambda x.N$ if and only if $M^s = \texttt{abs}^s \lceil x' \rceil\, N^s[\lceil x' \rceil / x]$,
- $M = PQ$ if and only if $M^s = \texttt{app}^s\, P^s\, Q^s$,

where x' is fresh, chosen in $\{\mathrm{x}_1, \dots, \mathrm{x}_s\}$.

Lemma 19. *Let $s > 0$. If M and N are linear λ-terms, then $M^s[N^s/x] = (M[N/x])^s$.*

Proof. By induction on $|M|$. If $M = x$ then $x^s[N^s/x] = x[N^s/x] = N^s = (x[N/x])^s$. If $M = PQ$ then either x occurs in P or it occurs in Q. Let us consider the case $x \in FV(P)$, the other case being similar: by using the induction hypothesis we have $(PQ)^s[N^s/x] = \texttt{app}^s\, P^s[N^s/x]\, Q^s = \texttt{app}^s (P[N/x])^s\, Q^s = ((P[N/x]Q)^s = ((PQ)[N/x])^s$. If $M = \lambda y.P$ then we have $(\lambda y.P)^s[N^s/x] = \texttt{abs}^s \lceil y' \rceil\, P^s[N^s/x][\lceil y' \rceil/y] = \texttt{abs}^s \lceil y' \rceil\, (P[N/x])^s[\lceil y' \rceil/y] = (\lambda y.P[N/x])^s = ((\lambda y.P)[N/x])^s$. $\square$

Lemma 20. *Let $s > 0$. If M is a linear λ-term in normal form such that $|M| \leq s$ with free variables $x_1, \dots, x_n$, then*

$$M^s[\lceil \vec{x'} \rceil] \to_\beta^* \lceil M[x'_1/x_1, \dots, x'_n/x_n] \rceil$$

where $\lceil \vec{x'} \rceil = [\lceil x'_1 \rceil/x_1, \dots, \lceil x'_n \rceil/x_n]$ and $x'_1, \dots, x'_n$ are distinct and fresh in $\{\mathrm{x}_1, \dots, \mathrm{x}_s\}$.

Proof. By induction on $|M|$. If $M = x$ then $\exists i \leq n\ x_i = x$, so that $x^s[\lceil x'_i \rceil/x] = x[\lceil x'_i \rceil/x] = \lceil x'_i \rceil = \lceil x[x'_i/x] \rceil$. If $M = \lambda y.N$ then, using the induction hypothesis, we have:

$$\begin{aligned}
(\lambda y.N)^s[\lceil \vec{x'} \rceil] &= (\texttt{abs}^s \lceil y' \rceil\, N^s[\lceil y' \rceil/y])[\lceil \vec{x'} \rceil] \\
&= \texttt{abs}^s \lceil y' \rceil\, N^s[\lceil \vec{x'} \rceil, \lceil y' \rceil/y] \\
&\to_\beta^* \texttt{abs}^s \lceil y' \rceil\, \lceil N[x'_1/x_1, \dots, x'_n/x_n, y'/y] \rceil \\
&\to_\beta^* \lceil \lambda y'.N[x'_1/x_1, \dots, x'_n/x_n, y'/y] \rceil && \text{Lem. 18} \\
&=_\alpha \lceil (\lambda y.N)[x'_1/x_1, \dots, x'_n/x_n] \rceil.
\end{aligned}$$

If $M = PQ$ then let $y_1, \dots, y_m$ (resp. $z_1, \dots, z_k$) be the free variables of P (resp. Q), and let $\vec{x'} = y'_1, \dots, y'_m, z'_1, \dots, z'_k$. Then we have:

$$\begin{aligned}
&(PQ)^s[\lceil \vec{x'} \rceil] = \\
&= \texttt{app}^s\, P^s[\lceil \vec{y'} \rceil]\, Q^s[\lceil \vec{z'} \rceil] \\
&\to_\beta^* \texttt{app}^s \lceil P[y'_1/y_1, \dots, y'_m/y_m] \rceil \lceil Q[z'_1/z_1, \dots, z'_k/z_k] \rceil \\
&\to_\beta^* \lceil P[y'_1/y_1, \dots, y'_m/y_m] Q[z'_1/z_1, \dots, z'_k/z_k] \rceil && \text{Lem. 18} \\
&= \lceil (PQ)[x'_1/x_1, \dots, x'_n/x_n] \rceil.
\end{aligned}$$

$\square$

Lemma 21. *Let M be a η-long normal form of type A with context $\Gamma = x_1 : A_1, \dots, x_n : A_n$, where A is a Π_1 type and Γ is a context of Σ_1 types, and let $\sum_{i=1}^m |A_i^-| + |A^-| = k$ and $s = c \cdot (k \cdot \log k)$, for some c large enough. Then:*

$$(\texttt{enc}_A^s\, M[\Gamma])[\lceil \vec{x'} \rceil] \to_\beta^* \lceil M[x'_1/x_1, \dots, x'_n/x_n] \rceil$$

where $\lceil \vec{x'} \rceil = [\lceil x'_1 \rceil/x_1, \dots, \lceil x'_n \rceil/x_n]$, with $x'_1, \dots, x'_n$ distinct and chosen fresh in $\{\mathrm{x}_1, \dots, \mathrm{x}_s\}$.

Proof. First, notice that, by Lemma 12, $\Gamma^-[\mathbf{B}^s] \vdash M : A^-[\mathbf{B}^s]$. By Lemma 20 it suffices to prove by induction on $|M|$ that the reduction $\mathtt{enc}^s_A\, M[\Gamma] \to^*_\beta M^s$ holds. If $M = x$ then $A = \alpha$ and $\Gamma = x : \alpha$, because M is in η-long normal form, so that we have $\mathtt{enc}^s_\alpha\, x[x : \alpha] = \mathtt{enc}^s_\alpha(\overline{\mathtt{enc}}^s_\alpha\, x) \to^*_\beta x = x^s$. If $M = \lambda y.N$ then $A = B \multimap C$, so that:

$$\begin{array}{ll}
\mathtt{enc}^s_{B\multimap C}((\lambda y.N)[\Gamma]) & \\
\quad \to_\beta \mathtt{abs}^s \ulcorner x' \urcorner(\mathtt{enc}^s_C((\lambda y.N[\Gamma])(\overline{\mathtt{enc}}^s_B \ulcorner y' \urcorner))) & \text{Def. 17} \\
\quad \to_\beta \mathtt{abs}^s \ulcorner y' \urcorner(\mathtt{enc}^s_C(N[\Gamma][\overline{\mathtt{enc}}^s_B \ulcorner y' \urcorner / y])) & \\
\quad = \mathtt{abs}^s \ulcorner y' \urcorner(\mathtt{enc}^s_C(N[\Gamma][\overline{\mathtt{enc}}^s_B\, x / x]))[\ulcorner y' \urcorner / y] & \\
\quad = \mathtt{abs}^s \ulcorner y' \urcorner(\mathtt{enc}^s_C(N[\Gamma, y : B]))[\ulcorner y' \urcorner / y] & \text{Def. 17} \\
\quad \to^*_\beta \mathtt{abs}^s \ulcorner y' \urcorner(N^s[\ulcorner y' \urcorner / y]) & \text{IH} \\
\quad = (\lambda y.N)^s. &
\end{array}$$

Last, suppose $M = P[yN/x]$, and let Σ, Δ be contexts such that $\Gamma = \Sigma, \Delta, y : B \multimap C$, $\mathrm{dom}(\Sigma) = FV(P)$, and $\mathrm{dom}(\Delta) = FV(N)$. Then we have:

$$\begin{array}{ll}
\mathtt{enc}^s_A(P[yN/x])[\Gamma] & \\
\quad = \mathtt{enc}^s_A(P[\Sigma][(yN)[\Delta, y : B \multimap C]/x]) & \\
\quad = \mathtt{enc}^s_A(P[\Sigma][(\overline{\mathtt{enc}}^s_{B\multimap C}\, y)N[\Delta]/x]) & \\
\quad \to^*_\beta \mathtt{enc}^s_A(P[\Sigma][\overline{\mathtt{enc}}^s_C(\mathtt{app}^s\, y(\mathtt{enc}^s_B\, N[\Delta]))/x]) & \text{Def. 17} \\
\quad \to^*_\beta \mathtt{enc}^s_A(P[\Sigma][\overline{\mathtt{enc}}^s_C(\mathtt{app}^s\, y\, N^s)/x]) & \text{IH} \\
\quad = \mathtt{enc}^s_A(P[\Sigma][\overline{\mathtt{enc}}^s_C(yN)^s/x]) & \\
\quad = \mathtt{enc}^s_A(P[\Sigma, x : C])[(yN)^s/x] & \text{Def. 17} \\
\quad \to^*_\beta P^s[(yN)^s/x] & \text{IH} \\
\quad = (P[yN/x])^s & \text{Lem. 19.}
\end{array}$$

□

3.2.3 The linear λ-term $\mathtt{dec}^s_A$

The linear λ-term $\mathtt{dec}^s_A$ is the component of $\mathtt{D}_A$ requiring the type inhabitation. Roughly, it takes a tuple of boolean values encoding the η-long normal form V_A of a ground type A in input, and it produces the pair $\langle V_A, V_A\rangle$. To ensure that $\mathtt{dec}^s_A$ is defined on all possible inputs, it is built in such a way that it returns a default inhabitant of A whenever the tuple of booleans in input does not encode any λ-term.

Definition 20 (The linear λ-term $\mathtt{dec}^s_A$)**.** Let A be a ground type and let V' be a value of type A. If for some c large enough $s = c \cdot (|A^-| \cdot \log |A^-|)$, then we define the linear λ-term $\mathtt{dec}^s_A : \mathbf{B}^s \multimap A \otimes A$ as follows:

$$\lambda x.\mathtt{if}\ x\ \mathtt{then}\ [P_{\mathtt{tt}^s}, P_{\langle \mathtt{tt}^{s-1}, \mathtt{ff}\rangle}, \ldots, P_{\langle \mathtt{ff}, \mathtt{tt}^{s-1}\rangle}, P_{\mathtt{ff}^s}]$$

where, for all $T = \langle \mathtt{b}_1, \ldots, \mathtt{b}_s\rangle$ of type $\mathbf{B}^s$:

$$P_T = \begin{cases} \langle V_A, V_A\rangle & \text{if } \langle \mathtt{b}_1, \ldots, \mathtt{b}_s\rangle = \ulcorner V_A \urcorner \\ \langle V', V'\rangle & \text{otherwise.} \end{cases}$$

We are now able to prove the fundamental result of this section:

Theorem 22 (Duplication [65]). *Every inhabited ground type is duplicable.*

Proof. The duplicator $\mathtt{D}_A$ of a inhabited ground type is defined as follows: we fix $s = c \cdot (|A^-| \cdot \log |A^-|)$, we fix a default value V' of A, and we set:

$$\mathtt{D}_A \triangleq \mathtt{dec}^s_A \circ \mathtt{enc}^s_A \circ \mathtt{sub}^s_A : A \multimap A \otimes A$$

where $M \circ N \triangleq \lambda z.M(Nz)$. By Lemma 15, Lemma 17, and Lemma 21, for all values V of type A, we have:

$$\mathtt{D}_A\, V \rightarrow^*_\beta \langle V_A, V_A \rangle \rightarrow^*_\eta \langle V, V \rangle.$$

□

Remark 4. If A is a ground type inhabited by the value V', we shall write $\mathtt{D}^{V'}_A$ to stress that the default inhabitant of A used in constructing the duplicator $\mathtt{D}_A$ of A is V'.

3.3 The system LEM and basic properties

In this section, we present the *Linearly Exponential Multiplicative Type Assignment*, LEM for short, a new system internalizing the mechanisms of weakening and contraction that exist in IMLL_2 (Theorems 9 and 10). It is defined as a type assignment for the term calculus $\Lambda_{l,\downarrow}$ that we obtain by endowing Λ_l with explicit and type-dependent constructs `copy` and `discard` performing duplication and erasure in a limited way. LEM extends IMLL_2 with inference rules for the modality "$\downarrow$" that closely recall the exponential rules in Linear Logic.

We explore both the computational and the proof-theoretical properties of the system. First, we define special cut-elimination steps, called *lazy*, that prevent undesired forms of duplication and erasure of derivations. The lazy cut-elimination is too weak, since we may run into deadlock situations, i.e. instances of *cut* that cannot be ruled out. We then show a relevant class of types, called *lazy types*, whose derivations can always be rewritten into cut-free ones by applying a specific lazy cut-elimination strategy (Theorem 29).

Thanks to a careful positioning of "$\downarrow$" in the types of LEM, essentially as done in [40], we are able to prove the Subject reduction property (Theorem 32).

Finally, we define a translation from LEM into IMLL_2 and we prove a simulation result (Theorem 35) showing how the constructs `copy` and `discard` in the former system relate to the duplicators and the erasers we studied in the latter. One of the striking aspects of the translation is Theorem 37, stating essentially that the construct `copy` exponentially compresses the mechanism of linear duplication of IMLL_2.

3.3.1 The system LEM

The types of LEM are built from the linear implication "$\multimap$", the second-order quantifier "$\forall$", and the new modality "$\downarrow$" that applies only to closed types free from negative occurrences of $\forall$. These latter types are the representatives in LEM of the ground types in Definition 10, and will be the only ones the system is allowed to explicitly weaken and contract.

Definition 21 (Types of LEM). Let $\mathcal{X}$ be a denumerable set of type variables. The following grammar (3.11) generates the *linear types*, while the grammar (3.12) generates the *exponential types*:

$$A := \alpha \mid \sigma \multimap A \mid \forall \alpha.A \tag{3.11}$$
$$\sigma := A \mid \downarrow\sigma \tag{3.12}$$

where $\alpha \in \mathcal{X}$ and, in the last clause of the grammar (3.12), i.e. the one introducing $\downarrow\sigma$, *the type σ must be closed and without negative occurrences of* $\forall$. The set of all types generated by the grammar (3.12) will be denoted $\Theta_{\downarrow}$. A type is *strictly exponential* if it is of the form $\downarrow\sigma$. A *strictly exponential context* is a context containing only strictly exponential types and, similarly, a *linear context* contains only linear types. If Γ is $x_1 : A_1, \dots, x_n : A_n$, then $\downarrow\Gamma$ is $x_1 : \downarrow A_1, \dots, x_n : \downarrow A_n$. The standard meta-level substitution of B for every occurrence of α in A is denoted $A\langle B/\alpha\rangle$. Finally, the *size* of a type σ in $\Theta_{\downarrow}$, written $|\sigma|$, is the number of nodes in the syntax tree of σ.

Remark 5. Syntactically replacing the Linear Logic modality "!" for "$\downarrow$" in (3.11) and (3.12) yields a subset of Gaboardi and Ronchi's *essential types* [40], introduced to prove Subject reduction in a variant of Soft Linear Logic [56]. Essential types forbid the occurrences of modalities in the right-hand side of an implication, such as in $A \multimap !B$ (see Section 4.2.1).

We shall define LEM as a type assignment for the term calculus $\Lambda_{l,\downarrow}$, which is essentially the standard linear λ-calculus with two type-dependent constructs for erasure and duplication, $\mathtt{discard}_\sigma$ and $\mathtt{copy}^V_\sigma$, the latter being also decorated with a value V, i.e. a closed and normal linear λ-term. Values, will be the only terms these new constructs are able to copy and erase.

Definition 22 (The calculi $\Lambda_{\downarrow}$ and $\Lambda_{l,\downarrow}$)**.**

- Let $\mathcal{V}$ be a denumerable set of variables. The set $\Lambda_{\downarrow}$ of *terms* is generated by the following grammar:

$$M := x \mid \lambda x.M \mid MM \mid \mathtt{discard}_\sigma\, M \;\mathtt{in}\; M \mid \mathtt{copy}^V_\sigma\, M \;\mathtt{as}\; x, y \;\mathtt{in}\; M \tag{3.13}$$

  where $x, y \in \mathcal{V}$, V is a value (Definition 7), and $\sigma \in \Theta_{\downarrow}$. The set of the free variables of a term, and the notion of size are standard for variables, abstractions, and applications. Their extensions to the new constructors are:

$$\begin{aligned} FV(\mathtt{discard}_\sigma\, M \;\mathtt{in}\; N) &= FV(M) \cup FV(N) \\ FV(\mathtt{copy}^V_\sigma\, M \;\mathtt{as}\; x, y \;\mathtt{in}\; N) &= FV(M) \cup (FV(N) \setminus \{x, y\}) \\ |\mathtt{discard}_\sigma\, M \;\mathtt{in}\; N| &= |M| + |N| + 1 \\ |\mathtt{copy}^V_\sigma\, M \;\mathtt{as}\; x, y \;\mathtt{in}\; N| &= |M| + |N| + |V| + 1. \end{aligned} \tag{3.14}$$

  The meta-level substitution of N for the free occurrences of x in M, written $M[N/x]$, is defined as usual. A *context* is a term containing a unique hole $[\cdot]$, generated by the following grammar:

$$\begin{aligned} \mathcal{C} := [\cdot] \mid \lambda x.\mathcal{C} \mid \mathcal{C}M \mid M\mathcal{C} \mid \mathtt{discard}_\sigma\, \mathcal{C} \;\mathtt{in}\; M \mid \mathtt{discard}_\sigma\, M \;\mathtt{in}\; \mathcal{C} \\ \mathtt{copy}^V_\sigma\, \mathcal{C} \;\mathtt{as}\; x, y \;\mathtt{in}\; M \mid \mathtt{copy}^V_\sigma\, M \;\mathtt{as}\; x, y \;\mathtt{in}\; \mathcal{C} \end{aligned} \tag{3.15}$$

  where, given a context $\mathcal{C}$ and a term M, $\mathcal{C}[M]$ denotes the term obtained by substituting the unique hole in $\mathcal{C}$ with M allowing the possible capture of free variables of M.

- The *one-step reduction relation* $\rightarrow$ is a binary relation over $\Lambda_{\downarrow}$. It is defined by the reduction rules in Figure 3.4(a) and by the commuting conversions in Figure 3.4(b), and applies in any context generated by (3.15). Its reflexive and transitive closure is denoted $\rightarrow^*$. A term is said a (or is in) *normal form* if no reduction applies to it.

- A term $M \in \Lambda_{\downarrow}$ is *linear* if:

$$\begin{array}{rcl}
(\lambda x.M)N & \to & M[N/x] \\
\texttt{discard}_\sigma\, V \texttt{ in } M & \to & M \\
\texttt{copy}^{V'}_\sigma\, V \texttt{ as } x,y \texttt{ in } M & \to & M[V/x, V/y]
\end{array}$$

(a) Reduction rules.

$$\begin{array}{rcl}
(\texttt{discard}_\sigma\, M \texttt{ in } N)P & \to & \texttt{discard}_\sigma\, M \texttt{ in } (NP) \\
\texttt{discard}_\sigma\, (\texttt{discard}_\tau\, M \texttt{ in } N) \texttt{ in } P & \to & \texttt{discard}_\tau\, M \texttt{ in } (\texttt{discard}_\sigma\, N \texttt{ in } P) \\
\texttt{copy}^{V}_\sigma\, (\texttt{discard}_\tau\, M \texttt{ in } N) \texttt{ as } x,y \texttt{ in } P & \to & \texttt{discard}_\tau\, M \texttt{ in } (\texttt{copy}^{V}_\sigma\, N \texttt{ as } x,y \texttt{ in } P) \\
(\texttt{copy}^{V}_\sigma\, M \texttt{ as } x,y \texttt{ in } N)P & \to & \texttt{copy}^{V}_\sigma\, M \texttt{ as } x,y \texttt{ in } (NP) \\
\texttt{discard}_\sigma\, (\texttt{copy}^{V}_\tau\, M \texttt{ as } x,y \texttt{ in } N) \texttt{ in } P & \to & \texttt{copy}^{V}_\tau\, M \texttt{ as } x,y \texttt{ in } (\texttt{discard}_\sigma\, N \texttt{ in } P) \\
\texttt{copy}^{V'}_\sigma\, (\texttt{copy}^{V}_\tau\, M \texttt{ as } x,y \texttt{ in } N) \texttt{ as } z,v \texttt{ in } P & \to & \texttt{copy}^{V}_\tau\, M \texttt{ as } x,y \texttt{ in } (\texttt{copy}^{V'}_\sigma\, N \texttt{ as } z,v \texttt{ in } P)
\end{array}$$

(b) Commuting conversions.

Figure 3.4: Reduction rules and commuting conversions for $\Lambda_\downarrow$.

- each free variable of M has just one occurrence free in it;
- for each subterm $\lambda x.N$ of M, x occurs in N exactly once;
- for each subterm $\texttt{copy}^{V}_\sigma\, M' \texttt{ as } y,z \texttt{ in } N$ of M, y and z occur in N exactly once.

The set of all linear terms in $\Lambda_\downarrow$ is denoted $\Lambda_{l,\downarrow}$.

Observe that $\Lambda_l \subseteq \Lambda_{l,\downarrow}$. Hence, each value is a linear term in $\Lambda_{l,\downarrow}$.

Proposition 23. *If $M \in \Lambda_{l,\downarrow}$ and $M \to N$ then $N \in \Lambda_{l,\downarrow}$.*

Proof. Straightforward. □

The type assignment system LEM makes both the type and the term annotations in the constructs $\texttt{discard}_\sigma$ and $\texttt{copy}^{V}_\sigma$ meaningful, and its definition is driven by the structure of the types in Definition 21.

Definition 23 (The system LEM)**.** LEM is the type assignment system (in sequent style) for the term calculus $\Lambda_{l,\downarrow}$ displayed in Figure 3.5. It extends IMLL_2 in Figure 3.1(a) with the rules *promotion* p, *dereliction* d, *weakening* w and *contraction* c. As usual, $\multimap$R, $\forall$R, and p are right rules while $\multimap$L, $\forall$L, d, w, and c are left ones.

The rule *ax* in Figure 3.5 cannot introduce exponential types, like in the type assignment systems for the *essential types* [40]. This is the key observation for proving:

Proposition 24. *If $\mathcal{D} \lhd \Gamma \vdash M : \downarrow\sigma$ is a derivation in* LEM*, then Γ is a strictly exponential context.*

$$\frac{}{x : A \vdash x : A}\ ax \qquad \frac{\Gamma \vdash N : \sigma \qquad \Delta, x : \sigma \vdash M : \tau}{\Gamma, \Delta \vdash M[N/x] : \tau}\ cut$$

$$\frac{\Gamma, x : \sigma \vdash M : B}{\Gamma \vdash \lambda x.M : \sigma \multimap B}\ \multimap\mathrm{R} \qquad \frac{\Gamma \vdash N : \sigma \qquad \Delta, x : B \vdash M : \tau}{\Gamma, \Delta, y : \sigma \multimap B \vdash M[yN/x] : \tau}\ \multimap\mathrm{L}$$

$$\frac{x_1 : \downarrow\sigma_1, \ldots, x_n : \downarrow\sigma_n \vdash M : \sigma}{x_1 : \downarrow\sigma_1, \ldots, x_n : \downarrow\sigma_n \vdash M : \downarrow\sigma}\ p \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma, y : \downarrow\sigma \vdash M[y/x] : \tau}\ d$$

$$\frac{\Gamma \vdash M : \tau}{\Gamma, x : \downarrow\sigma \vdash \mathtt{discard}_\sigma\ x\ \mathtt{in}\ M : \tau}\ w \qquad \frac{\Gamma, y : \downarrow\sigma, z : \downarrow\sigma \vdash M : \tau \qquad \vdash V : \sigma}{\Gamma, x : \downarrow\sigma \vdash \mathtt{copy}^V_\sigma\ x\ \mathtt{as}\ y, z\ \mathtt{in}\ M : \tau}\ c$$

$$\frac{\Gamma \vdash M : A\langle\gamma/\alpha\rangle \qquad \gamma \notin FV(\Gamma)}{\Gamma \vdash M : \forall\alpha.A}\ \forall\mathrm{R} \qquad \frac{\Gamma, x : A\langle B/\alpha\rangle \vdash M : \tau}{\Gamma, x : \forall\alpha.A \vdash M : \tau}\ \forall\mathrm{L}$$

Figure 3.5: The system LEM.

Proof. By structural induction on the derivation $\mathcal{D}$. □

Let us remark that the rules p, d, w, c of LEM are reminiscent of the namesake Linear Logic exponential rules, but they only apply to types $\downarrow\sigma$ generated by (3.12), i.e. such that σ is closed and free from negative occurrences of $\forall$. Intuitively, a type σ appearing in $\downarrow\sigma$ can be seen as the representative in LEM of a ground type (Definition 10). Then, by weakening and contracting the strictly exponential type $\downarrow\sigma$, the rules w and c of LEM mirror the linear contraction and weakening of a ground type of IMLL_2. In particular, the rule c contracting $\downarrow\sigma$ has one premise more than the contraction rule in Linear Logic. This premise "witnesses" that the type σ is inhabited by at least one value V, which is required to faithfully express in LEM the linear contraction mechanism of IMLL_2, as Theorem 10 states that a duplicator exists for those ground types which are *inhabited*. In Section 3.3.4, these intuitions will be formalized by introducing a translation of LEM into IMLL_2 (Definition 30) that shows how the construct $\mathtt{copy}^V_\sigma$ exponentially compresses the linear duplication mechanism encoded in a duplicator (Theorem 37).

A similar reasoning applies to the construct $\mathtt{discard}_\sigma$, that expresses the eraser of a ground type in IMLL_2. Since duplicators and erasers of a ground type apply to the set of values that inhabit it, we designed the reduction rules in Figure 3.4(a) in such a way that the constructs $\mathtt{copy}^V_\sigma$ and $\mathtt{discard}_\sigma$ copy and erase values only.

We conclude by commenting about how "$\downarrow$" and LL's "!" differ. Intuitively, the latter allows to duplicate or erase logical structure, or terms, *at once*, which is the standard way to computationally interpret contraction and weakening of a logical system. The modality "$\downarrow$" identifies duplication and erasure processes with a more *constructive* nature. The duplication proceeds step-by-step among a whole set of possible choices in order to identify those ones that cannot contain the copies of the term we are interested to duplicate, until it eventually reaches what it searches. Then, it exploits erasure. Erasing means eroding step-by-step a derivation or a term, according to the type that drives its construction.

3.3.2 Cut-elimination and its cubical complexity

LEM is a type assignment for the linear calculus $\Lambda_{l,\downarrow}$ whose reduction rules depicted in Figure 3.4(a) limit duplication and erasure to values. These reduction rules are more restrictive than the cut-elimination steps that we could perform on LEM. Indeed, once replaced "!" for "$\downarrow$" and forgetting the term annotations, the cut-elimination of LEM would correspond essentially to the one of IMELL_2 (see Section 2.2.2). So, for example, in the following derivation of LEM:

$$\dfrac{\dfrac{\dfrac{\vdots}{y : \downarrow A, z : \downarrow \mathbf{1} \vdash yz : \mathbf{1}}}{y : \downarrow A, z : \downarrow \mathbf{1} \vdash yz : \downarrow\mathbf{1}}\, p \qquad \dfrac{\dfrac{\vdots}{\Gamma, x_1 : \downarrow\mathbf{1}, x_2 : \downarrow\mathbf{1} \vdash M : \tau} \qquad \dfrac{\vdots}{\vdash \mathbf{I} : \mathbf{1}}}{\Gamma, x : \downarrow\mathbf{1} \vdash \mathtt{copy}^{\mathbf{I}}_{\mathbf{1}}\, x\ \mathtt{as}\ y, z\ \mathtt{in}\ M : \tau}\, c}{\Gamma, y : \downarrow A, z : \downarrow\mathbf{1} \vdash \mathtt{copy}^{\mathbf{I}}_{\mathbf{1}}\, yz\ \mathtt{as}\ y, z\ \mathtt{in}\ M : \tau}\, cut \tag{3.16}$$

the typical cut-elimination step moves the *cut* upward. But this would duplicate the *open* term yz, erroneously yielding a non-linear term. Hence, at the proof-theoretical level, cut-elimination steps exist that cannot correspond to any reduction on terms. In order to circumvent the here above misalignment, we proceed as follows:

- We define the *lazy cut-elimination steps*, ruling out any attempt to eliminate an instance of *cut* like (3.16). The drawback of this approach is that we may run into *deadlocks*, i.e. into instances of *cut* that cannot be eliminated.
- We define a relevant class of types, called *lazy*, and we show that a *lazy cut-elimination strategy* exists that is able to eliminates all the cut rules that may occur in a derivation of a lazy type. The cost of the elimination is cubical (Theorem 29).

Definition 24 (Cut rules of LEM)**.** Let (X, Y) identify an instance:

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash N : \sigma}\, X \qquad \dfrac{\vdots}{\Delta, x : \sigma \vdash M : \tau}\, Y}{\Gamma, \Delta \vdash M[N/x] : \tau}\, cut \tag{3.17}$$

of the rule *cut* that occurs in a given derivation $\mathcal{D}$ of LEM, where X and Y are two of the rules in Figure 3.5. *Axiom cuts* involve ax, and are of the form (X, ax) or (ax, Y), for some X and Y. *Exponential cuts* are (p,d), (p,w), and (p,c). *Principal cuts* are $(\multimap\text{R}, \multimap\text{L})$, $(\forall\text{R}, \forall\text{L})$ and every exponential cut. *Symmetric cuts* contain axiom and principal cuts. Every symmetric cut that is not exponential is *multiplicative*. *Commuting cuts* are all the remaining instances of *cut*, not mentioned here above, (p, p) included, for example.

A *lazy cut* is every instance of the cut rule (3.17) which is both exponential and such that N is a value.

A *deadlock* is every instance of the cut rule (3.17) which is both exponential and such that $\Gamma \neq \emptyset$. Otherwise, it is *safe*.

The lazy cut-elimination rules that we introduce here below are the standard ones, but restricted to avoid the elimination of non-lazy instances of the exponential cuts (p, d), (p, w) and (p, c).

Definition 25 (Lazy cut-elimination rules)**.** Figure 3.6 introduces the *lazy cut-elimination rules* for the principal cuts. The elimination rules for commuting and axiom cuts are standard, so we omit them all from Figure 3.6; the (possibly) less obvious commuting ones can be recovered from the reductions on terms in Figure 3.4(b). We remark that the elimination of the principal cuts

($\forall$R, $\forall$L) and (p, d) does not modify the subject of their concluding judgment. So, we call them *insignificant* as every other cut-elimination rule non influencing their concluding subject. Given a derivation $\mathcal{D}$, we write $\mathcal{D} \rightsquigarrow \mathcal{D}'$ if $\mathcal{D}$ rewrites to some $\mathcal{D}$ by a lazy cut-elimination rule.

The introduction of the lazy cut-elimination rules is a way of preventing the erasure and the duplication of terms that are not values, and hence to restore a correspondence between cut-elimination and term reduction. However, we can run into derivations containing exponential cuts that will never turn into lazy cuts, like the deadlock in (3.16). The solution we adopt is to identify a set of judgments whose derivations can be rewritten into cut-free ones by a sequence of lazy cut-elimination steps.

Definition 26 (Lazy types, lazy judgments and lazy derivations)**.** We say that σ is a *lazy type* if it contains no *negative* occurrences of $\forall$. Also, we say that $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$ is a *lazy judgment* if τ is a lazy type and $\sigma_1, \ldots, \sigma_n$ contain no *positive* occurrences of $\forall$. Last, $\mathcal{D} \triangleleft \Gamma \vdash M : \tau$ is called a *lazy derivation* if $\Gamma \vdash M : \tau$ is a lazy judgment.

Lemma 25 and 26 here below, as well as Definition 27 and 28, are the last preliminaries to show the relevance of lazy cuts that occur in lazy derivations.

Lemma 25.

(1) Every type of the form $\downarrow\sigma$ is closed and lazy.

(2) Every closed type has at least a positive quantification.

(3) Let R be any instance of $\forall$L, d, w, c, and p, the latter with a non empty context. The conclusion of R is not lazy.

(4) Let R be any instance of ax, $\multimap$ R, $\multimap$ L, $\forall$R, and p, the latter with an empty context. If the conclusion of R is lazy, then, every premise of R is lazy.

(5) If $\mathcal{D}$ is a cut-free and lazy derivation of LEM*, then all its judgments are lazy and no occurrence of $\forall$L, d, w, c, and p, the latter with a non empty context, can exist in $\mathcal{D}$.*

Proof. Point (1) holds by Definition 21. Point (2) is by a structural induction on types. Concerning point (3), the conclusions of d, w, c, and p contain $\downarrow\sigma$. This is a closed type, hence, by point (2), such conclusions are not lazy judgments. Moreover, $\forall$L introduces a positive occurrence of $\forall$ in the context of its conclusion, so that the latter cannot be a lazy judgment. Point (4) is a case analysis on every listed inference rule. As for point (5), we can proceed by structural induction on $\mathcal{D}$. By definition, the conclusion of $\mathcal{D}$ is a lazy judgment. Point (3) excludes that one among $\forall$L, d, w, c, and p (with a non empty context) may be the last rule of $\mathcal{D}$. So, only one among ax, $\multimap$R, $\multimap$L, $\forall$R, and p (with an empty context) can be the concluding rule, say R, of $\mathcal{D}$. Point (4) implies that all premises of R are lazy. Hence, we can apply the inductive hypothesis and conclude. □

The size of a derivation is usually defined as the number of rules of that derivation. For ease of presentation, we shall define a slightly different notion of size for derivations in LEM, which does not affect the fundamental result of this subsection (Theorem 29):

Definition 27 (Size of a derivation)**.** The *size* $|\mathcal{D}|$ of a derivation $\mathcal{D}$ in LEM is defined by induction:

- If $\mathcal{D}$ is ax then $|\mathcal{D}| = 1$;
- if $\mathcal{D}$ is a derivation $\mathcal{D}'$ that concludes by a rule with a single premise, then $|\mathcal{D}| = |\mathcal{D}'| + 1$;

$$\dfrac{\dfrac{\overset{\mathcal{D}}{\Gamma, x:\sigma \vdash M:A}}{\Gamma \vdash \lambda x.M : \sigma \multimap A}\multimap R \qquad \dfrac{\overset{\mathcal{D}'}{\Delta \vdash N:\sigma} \qquad \overset{\mathcal{D}''}{\Sigma, z:A \vdash P:\tau}}{\Delta,\Sigma, y:\sigma\multimap A \vdash P[yN/z]:\tau}\multimap L}{\Gamma,\Delta,\Sigma \vdash P[(\lambda x.M)N/z]:\tau}\mathit{cut} \rightsquigarrow \dfrac{\dfrac{\overset{\mathcal{D}'}{\Delta\vdash N:\sigma}\qquad \overset{\mathcal{D}}{\Gamma,x:\sigma\vdash M:A}}{\Gamma,\Delta\vdash M[N/x]:A}\mathit{cut} \qquad \overset{\mathcal{D}''}{\Sigma,z:A\vdash P:\tau}}{\Gamma,\Delta,\Sigma\vdash P[M[N/x]/z]:\tau}\mathit{cut}$$

$$\dfrac{\dfrac{\overset{\mathcal{D}}{\Gamma\vdash M:A\langle\gamma/\alpha\rangle}}{\Gamma\vdash M:\forall\alpha.A}\forall R \qquad \dfrac{\overset{\mathcal{D}'}{\Delta,x:A\langle B/\alpha\rangle\vdash N:\tau}}{\Delta,x:\forall\alpha.A\vdash N:\tau}\forall L}{\Gamma,\Delta\vdash N[M/x]:\tau}\mathit{cut} \rightsquigarrow \dfrac{\overset{\mathcal{D}\langle B/\alpha\rangle}{\Gamma\vdash M:A\langle B/\alpha\rangle} \qquad \overset{\mathcal{D}'}{\Delta,x:A\langle B/\alpha\rangle\vdash N:\tau}}{\Gamma,\Delta\vdash N[M/x]:\tau}\mathit{cut}$$

$$\dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V:\sigma}}{\vdash V:{\downarrow}\sigma}p \qquad \dfrac{\overset{\mathcal{D}'}{\Gamma,x:\sigma\vdash M:\tau}}{\Gamma,y:{\downarrow}\sigma\vdash M[y/x]:\tau}d}{\Gamma\vdash M[V/x]:\tau}\mathit{cut} \rightsquigarrow \dfrac{\overset{\mathcal{D}}{\vdash V:\sigma}\qquad \overset{\mathcal{D}'}{\Gamma,x:\sigma\vdash M:\tau}}{\Gamma\vdash M[V/x]:\tau}\mathit{cut}$$

$$\dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V:\sigma}}{\vdash V:{\downarrow}\sigma}p \qquad \dfrac{\overset{\mathcal{D}'}{\Delta\vdash M:\tau}}{\Delta,x:{\downarrow}\sigma\vdash \mathtt{discard}_\sigma\,x\ \mathtt{in}\ M:\tau}w}{\Delta\vdash \mathtt{discard}_\sigma\,V\ \mathtt{in}\ M:\tau}\mathit{cut} \rightsquigarrow \overset{\mathcal{D}'}{\Delta\vdash M:\tau}$$

$$\dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V:\sigma}}{\vdash V:{\downarrow}\sigma}p \qquad \dfrac{\overset{\mathcal{D}_1}{\Delta,y:{\downarrow}\sigma,z:{\downarrow}\sigma\vdash M:\tau}\qquad \overset{\mathcal{D}_2}{\vdash V':\sigma}}{\Delta,x:{\downarrow}\sigma\vdash \mathtt{copy}^{V'}_\sigma\,x\ \mathtt{as}\ y,z\ \mathtt{in}\ M:\tau}c}{\Delta\vdash \mathtt{copy}^{V'}_\sigma\,V\ \mathtt{as}\ y,z\ \mathtt{in}\ M:\tau}\mathit{cut} \rightsquigarrow \dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V:\sigma}}{\vdash V:{\downarrow}\sigma}p \qquad \dfrac{\dfrac{\overset{\mathcal{D}}{\vdash V:\sigma}}{\vdash V:{\downarrow}\sigma}p \qquad \overset{\mathcal{D}_1}{\Delta,y:{\downarrow}\sigma,z:{\downarrow}\sigma\vdash M:\tau}}{\Delta,z:{\downarrow}\sigma\vdash M[V/y]:\tau}\mathit{cut}}{\Delta\vdash M[V/y,V/z]:\tau}\mathit{cut}$$

Figure 3.6: *Lazy* cut-elimination rules for the principal cuts ($\multimap$R, $\multimap$L), ($\forall$L, $\forall$R), (p,d), (p,w), and (p,c).

- if $\mathcal{D}$ composes two derivations $\mathcal{D}'$ and $\mathcal{D}''$ by a rule with two premises, but different from c, then $|\mathcal{D}| = |\mathcal{D}'| + |\mathcal{D}''| + 1$;
- if $\mathcal{D}$ composes two derivations $\mathcal{D}'$ and $\mathcal{D}''$ by the rule c, then $|\mathcal{D}| = |\mathcal{D}'| + |\mathcal{D}''| + 3$.

Lemma 26. *Let $\mathcal{D} \lhd x_1 : \sigma_1, \dots, x_n : \sigma_n \vdash M : \sigma$ be a cut-free and lazy derivation:*

(1) M is a linear λ-term in normal form;

(2) $|M| \leq \sum_{i=1}^{n} |\sigma_i| + |\sigma|$;

(3) $|\mathcal{D}| = |M| + k$, where k is the number of $\forall$ and $\downarrow$ occurring in $\sigma_1, \dots, \sigma_n, \sigma$;

(4) if $\mathcal{D}' \lhd x_1 : \sigma_1, \dots, x_n : \sigma_n \vdash N : \sigma$ is a lazy and cut-free derivation, then $|N| \leq |M|$ implies $|\mathcal{D}'| \leq |\mathcal{D}|$;

(5) the set of values with lazy type σ is finite.

Proof. The assumptions on $\mathcal{D}$ allow to apply Lemma 25.(5) which implies that every judgment in $\mathcal{D}$ is lazy and free from $\forall$L, d, w, c and p (with a non empty context). Hence, we can prove points (1)-(3) by induction on the structure of $\mathcal{D}$. Point (4) is a corollary of point (3). Point (5) is a corollary of point (2). □

Definition 28 (Height of an inference rule)**.** Let $\mathcal{D} \lhd \Gamma \vdash M : \sigma$ be a derivation and R a rule instance in it. The *height of* R, written $h(R)$, is the number of rule instances from the conclusion $\Gamma \vdash M : \sigma$ of $\mathcal{D}$ upward to the conclusion of R. The *height of* $\mathcal{D}$, written $h(\mathcal{D})$, is the largest $h(R)$ among its rule instances.

Lemma 27 and 28 assure that we can eliminate lazy cuts from a lazy derivation.

Lemma 27 (Existence of a safe cut)**.** *Let $\mathcal{D}$ be a lazy derivation with only exponential cuts in it. At least one of those cuts is* safe*.*

Proof. Let $\Gamma \vdash M : \tau$ be the conclusion of $\mathcal{D}$. By contradiction, let us suppose that every occurrence of exponential cut in $\mathcal{D}$ is a deadlock. At least one of them, say C_m, has minimal height $h(C_m)$, i.e. no other *cut* occurs in the sequence of rule instances, say $R_1, \dots R_n$, from the conclusion of C_m down to the one of $\mathcal{D}$. Since C_m is a deadlock, its leftmost premise has form $\Delta \vdash N : \downarrow\sigma$, where $\Delta \neq \emptyset$. By Proposition 24, Δ is strictly exponential and hence $\Delta \vdash N : \downarrow\sigma$ is a non lazy judgment by Lemma 25.(1) and Lemma 25.(2). Lemma 25.(3) and the contraposition of Lemma 25.(4) imply that the non lazy judgment in C_m can only be transformed to a non lazy judgment by every R_i, with $1 \leq i \leq n$, letting the conclusion of $\mathcal{D}$ non lazy, so contradicting the assumption. Hence, C_m must be safe. □

Lemma 28 (Eliminating a lazy cut)**.** *Let $\mathcal{D}$ be a lazy derivation with only exponential cuts in it. A lazy derivation $\mathcal{D}^*$ exists such that both $\mathcal{D} \rightsquigarrow \mathcal{D}^*$, by reducing a lazy cut, and $|\mathcal{D}^*| < |\mathcal{D}|$.*

Proof. Lemma 27 implies that $\mathcal{D}$ contains at least an exponential cut which is safe. Let us take (p, X) with maximal height $h((p, X))$ among those safe instances of *cut*. So, if (p, X) has form:

$$\dfrac{\dfrac{\dfrac{\mathcal{D}'}{\vdash N : \sigma}}{\vdash N : \downarrow\sigma}\, p \qquad \dfrac{\vdots}{\Delta, x : \downarrow\sigma \vdash M : \tau}\, X}{\Delta \vdash M[N/x] : \tau}\, cut \tag{3.18}$$

then $\mathcal{D}'$ is a lazy derivation because $\downarrow\sigma$ is a lazy type by Lemma 25.(1). Since $\mathcal{D}'$ is lazy and can only contain exponential cuts, by Lemma 27 and by maximality of $h((p, X))$, it is forcefully cut-free. So, by Lemma 26.(1), we have that N is a value, i.e. (p, X) is a lazy cut and we can reduce it to obtain $\mathcal{D}^*$. If X in (3.18) is d or w, then it is simple to show that $|\mathcal{D}^*| < |\mathcal{D}|$. Let X be c. Then, (3.18) is:

$$\dfrac{\dfrac{\overset{\mathcal{D}'}{\vdash V:\sigma}}{\vdash V:\downarrow\sigma}\,p \qquad \dfrac{\overset{\mathcal{D}''}{\Delta, y:\downarrow\sigma, z:\downarrow\sigma \vdash M':\tau} \qquad \overset{\mathcal{D}'''}{\vdash V':\sigma}}{\Delta, x:\downarrow\sigma \vdash \mathtt{copy}^{V'}_{\sigma}\ x\ \mathtt{as}\ y,z\ \mathtt{in}\ M' : \tau}\,c}{\Delta \vdash \mathtt{copy}^{V'}_{\sigma}\ V\ \mathtt{as}\ y,z\ \mathtt{in}\ M':\tau}\,\mathit{cut} \tag{3.19}$$

with $\mathcal{D}'''$ lazy and cut-free for the same reasons as $\mathcal{D}'$ is. So, (3.19) can reduce to:

$$\dfrac{\dfrac{\overset{\mathcal{D}'}{\vdash V:\sigma}}{\vdash V:\downarrow\sigma}\,p \qquad \dfrac{\dfrac{\overset{\mathcal{D}'}{\vdash V:\sigma}}{\vdash V:\downarrow\sigma}\,p \qquad \overset{\mathcal{D}''}{\Delta, y:\downarrow\sigma, z:\downarrow\sigma \vdash M':\tau}}{\Delta, z:\downarrow\sigma \vdash M'[V/y]:\tau}\,\mathit{cut}}{\Delta \vdash M'[V/y, V/z]:\tau}\,\mathit{cut} \tag{3.20}$$

By Lemma 26.(5), we can safely assume that V' is a value with largest size among values of type σ. By Lemma 26.(4), from $|V| \leq |V'|$ we have $|\mathcal{D}'| \leq |\mathcal{D}'''|$. By applying Definition 27 to (3.19) and (3.20), we have $|\mathcal{D}^*| < |\mathcal{D}|$. ☐

Definition 29 (Lazy cut-elimination strategy)**.** Let $\mathcal{D}$ be a lazy derivation of LEM. We define a *round* on $\mathcal{D}$ as follows:

{1} Eliminate all the commuting instances of *cut*.

{2} If any instance of *cut* remains, it is necessarily symmetric. Reduce a multiplicative cut, if any. Otherwise, reduce a lazy exponential cut, if any.

The *lazy cut-elimination strategy* iterates rounds, starting from $\mathcal{D}$, whenever instances of *cut* exist in the obtained derivations.

Theorem 29 (Lazy cut-elimination has a cubic bound)**.** *Let $\mathcal{D}$ be a lazy derivation. The lazy cut-elimination reduces $\mathcal{D}$ to a cut-free $\mathcal{D}^*$ in $\mathcal{O}(|\mathcal{D}|^3)$ steps.*

Proof. Let $H(\mathcal{D})$ be the sum of the heights $h(\mathcal{D}')$ of all subderivations $\mathcal{D}'$ of $\mathcal{D}$ whose conclusion is an instance of *cut*. We proceed by induction on the lexicographically order of the pairs $(|\mathcal{D}|, H(\mathcal{D}))$. To show that the lazy cut-elimination strategy in Definition 29 terminates, we start by applying a round to $\mathcal{D}$, using step {1}. Every commuting cut-elimination step just moves an instance of *cut* upward, strictly decreasing $H(\mathcal{D})$ and leaving $|\mathcal{D}|$ unaltered. Let us continue by applying step {2} of the round. As usual, $|\mathcal{D}|$ shrinks when eliminating a multiplicative cut. If only exponential instances of *cut* remain, by Lemma 28 we can rewrite $\mathcal{D}$ to $\mathcal{D}'$ by reducing a lazy exponential cut in such a way that $|\mathcal{D}'| < |\mathcal{D}|$. Therefore, the lazy cut-elimination strategy terminates with a cut-free derivation $\mathcal{D}^*$.

We now exhibit a bound on the number of cut-elimination steps from $\mathcal{D}$ to $\mathcal{D}^*$. Generally speaking, we can represent a lazy strategy as:

$$\mathcal{D}=\mathcal{D}_0 \underbrace{\longrightarrow}_{cc_0} \mathcal{D}'_0 \rightsquigarrow \mathcal{D}_1 \cdots \underbrace{\longrightarrow}_{cc_i}\mathcal{D}'_i \rightsquigarrow \mathcal{D}_{i+1}\underbrace{\longrightarrow}_{cc_{i+1}}\cdots\ \mathcal{D}'_{n-1}\rightsquigarrow \mathcal{D}_n \underbrace{\longrightarrow}_{cc_n}\mathcal{D}'_n = \mathcal{D}^* \tag{3.21}$$

where every cc_j denotes the number of commuting cuts applied from derivation $\mathcal{D}_j$ to derivation $\mathcal{D}'_j$, for every $0 \leq j \leq n$. A bound on every cc_j is $|\mathcal{D}_j|^2$ because every instance of rule in $\mathcal{D}_j$ can, in principle, be commuted with every other. The first part of the proof implies $|\mathcal{D}_j| = |\mathcal{D}'_j|$, for every $0 \leq j \leq n$. Lemma 28 implies $|\mathcal{D}'_j| > |\mathcal{D}_{j+1}|$, for every $0 \leq j \leq n-1$. So, $n \leq |\mathcal{D}|$ and the total number of cut-elimination steps in (3.21) is $O(|\mathcal{D}| \cdot |\mathcal{D}|^2)$. □

Remark 6. The cubic bound on the lazy strategy keeps holding also when it is applied to *non-lazy* derivations. Of course, in that case, deadlocks may remain in the final derivation where no instance of *cut* can be further eliminated.

3.3.3 Subject reduction

The proof of the Subject reduction requires some well-known preliminary results.

Lemma 30 (Substitution). *If* $\Gamma \vdash M : \tau$ *then* $\Gamma[A/\alpha] \vdash M : \tau[A/\alpha]$*, for every* linear *type* A.

Proof. Straightforward. □

Lemma 31 (Generation).

(1) *If* $\mathcal{D} \lhd \Gamma \vdash \lambda x.M : \tau$*, then* $\tau = \downarrow^n \forall\vec{\alpha}.(\sigma \multimap A)$*, where* $\downarrow^n \triangleq \downarrow \overset{n}{\dots} \downarrow$ *and* $\vec{\alpha} = \alpha_1, \dots, \alpha_m$*, for some* $n \geq 0$ *and* $m \geq 0$.

(2) *If* $\mathcal{D} \lhd \Gamma \vdash M : \downarrow\sigma$*, then a derivation* $\mathcal{D}'$ *exists which is* $\mathcal{D}$ *with some rule permuted in order to get an instance of* p *as last rule of* $\mathcal{D}'$.

(3) *If* $\mathcal{D} \lhd \Gamma \vdash \lambda x.M : \forall\alpha.A$*, then a derivation* $\mathcal{D}'$ *exists which is* $\mathcal{D}$ *with some rule permuted in order to obtain an instance of* $\forall$R *as last rule of* $\mathcal{D}'$.

(4) *If* $\mathcal{D} \lhd \Gamma \vdash \lambda x.P : \sigma \multimap B$*, then a derivation* $\mathcal{D}'$ *exists which is* $\mathcal{D}$ *with some rule permuted in order to obtain an instance of* $\multimap$R *as last rule of* $\mathcal{D}'$.

(5) *If* $\mathcal{D} \lhd \Delta, x : \downarrow\sigma \vdash P[xN/y] : \tau$*, then* $\mathcal{D}$ *contains an application of* d *that introduces* $x : \downarrow\sigma$.

(6) *If* $\mathcal{D} \lhd \Delta, x : \forall\alpha.A \vdash P[xN/y] : \tau$*, then* $\mathcal{D}$ *contains an application of* $\forall$L *that introduces* $x : \forall\alpha.A$.

(7) *If* $\mathcal{D} \lhd \Delta, x : \sigma \multimap B \vdash P[xN/y] : \tau$*, then* $\mathcal{D}$ *contains an application of* $\multimap$L *that introduces* $x : \sigma \multimap B$.

(8) *If* $\mathcal{D} \lhd \Delta, x : \downarrow\sigma \vdash \mathtt{discard}_\sigma\, x\ \mathtt{in}\ P : \tau$*, then* $\mathcal{D}$ *contains an application of* w *that introduces* $x : \downarrow\sigma$.

(9) *If* $\mathcal{D} \lhd \Delta, x : \downarrow\sigma \vdash \mathtt{copy}^V_\sigma\, x\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ P : \tau$*, then* $\mathcal{D}$ *contains an application of* c *that introduces* $x : \downarrow\sigma$.

Proof. We can adapt the proof by Gaboardi and Ronchi in [40] to LEM because the types in Definition 21 are a subset of Gaboardi and Ronchi's *essential types*. In particular, point *(2)* relies on Proposition 24. □

Theorem 32 (Subject reduction). *If* $\Gamma \vdash M : \tau$ *and* $M \to M'$*, then* $\Gamma \vdash M' : \tau$.

Proof. We proceed by structural induction on $\mathcal{D}$ and case analysis of $M \to M'$. The crucial case is when $(\lambda x.P)Q$ exists in M and $\mathcal{D}$ contains:

$$\dfrac{\mathcal{D}' \lhd \Delta \vdash \lambda x.P : \sigma \qquad \mathcal{D}'' \lhd \Sigma, y:\sigma \vdash N[yQ/z]:\tau}{\mathcal{D} \lhd \Delta,\Sigma \vdash N[(\lambda x.P)Q/z]:\tau}\; cut$$

Lemma 68.(1) implies that $\sigma = \downarrow^n \forall\vec{\alpha}.(\sigma_1 \multimap C)$, where $\downarrow^n \triangleq \downarrow \overset{n}{\dots} \downarrow$ and $\vec{\alpha} = \alpha_1,\dots,\alpha_m$, for some $n \geq 0$ and $m \geq 0$. Lemma 68.(5)-(7) imply that $\mathcal{D}''$ has form:

$$\begin{array}{c} \dfrac{\begin{array}{c}\vdots\\ \Sigma_1''\vdash Q'':\sigma_1'\end{array} \qquad \begin{array}{c}\vdots\\ \Sigma_2'', z:C'\vdash N'':\tau''\end{array}}{\Sigma_1'',\Sigma_2'',y''':\sigma_1'\multimap C'\vdash N''[y'''Q''/z]:\tau''}\multimap\mathrm{L} \\ \vdots\,\gamma'' \\ \dfrac{\Sigma', y'':\forall\vec{\alpha}.(\sigma_1\multimap C)\vdash N'[y''Q'/z]:\tau'}{\Sigma',y':\downarrow\forall\vec{\alpha}.(\sigma_1\multimap C)\vdash N'[y'Q'/z]:\tau'}\; d \\ \vdots\,\gamma' \\ \Sigma, y:\downarrow^n\forall\vec{\alpha}.(\sigma_1\multimap C)\vdash N[yQ/z]:\tau \end{array} \tag{3.22}$$

where γ' and γ'' are sequences of applications of inference rules, $\sigma_1' = \sigma_1[A_1/\alpha_1,\dots,A_m/\alpha_m]$ and $C' = C[A_1/\alpha_1,\dots,A_m/\alpha_m]$, for some Σ', Σ_1'',Σ_2'',y',y'',y''', N',N'',Q',Q'', τ',τ'', A_1, …, A_m. Lemma 68.(2)-(4) imply that, permuting some of its rules, $\mathcal{D}'$ can be reorganized as:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Delta, x:\sigma_1 \vdash P : C\end{array}}{\Delta \vdash \lambda x.P : \sigma_1 \multimap C}\multimap\mathrm{R}}{\Delta \vdash \lambda x.P : \forall\vec{\alpha}.(\sigma_1 \multimap C)}\forall\mathrm{R}}{\Delta \vdash \lambda x.P : \downarrow^n\forall\vec{\alpha}.(\sigma_1 \multimap C)}\; p$$

where the concluding instances of p are necessary if $n > 0$ and can be applied since Δ is strictly exponential as a consequence of Proposition 24. Moreover, Lemma 30 assures that a derivation of $\Delta, x:\sigma_1' \vdash P : C'$ exists because $\alpha_1,\dots,\alpha_m$ are not free in Δ. Therefore:

$$\begin{array}{c} \dfrac{\dfrac{\begin{array}{c}\vdots\\ \Sigma_1''\vdash Q'':\sigma_1'\end{array} \qquad \begin{array}{c}\vdots\\ \Delta, x:\sigma_1'\vdash P:C'\end{array}}{\Delta,\Sigma_1''\vdash P[Q''/x]:C'}\; cut \qquad \begin{array}{c}\vdots\\ \Sigma_2'', z:C'\vdash N'':\tau''\end{array}}{\Delta,\Sigma_1'',\Sigma_2''\vdash N''[P[Q''/x]/z]:\tau''}\; cut \\ \vdots\,\gamma',\gamma'' \\ \Delta,\Sigma\vdash N[P[Q/x]/z]:\tau \end{array}$$

A similar proof exists, which relies on Lemma 68.(8), or Lemma 68.(9), when reducing $\texttt{discard}_\sigma$ V $\texttt{in}$ M, or $\texttt{copy}_\sigma^{V'}$ V $\texttt{as}$ y,z $\texttt{in}$ M. All the remaining cases concerning the commuting conversions are straightforward. □

3.3.4 Translation of LEM into IMLL$_2$ and exponential compression

The system LEM provides a logical status to copying and erasing operations that exist in IMLL$_2$. In what follows, we show that a translation $(\_)^\bullet$ from LEM into IMLL$_2$ exists that "unpacks" both the constructs $\texttt{discard}_\sigma$ and $\texttt{copy}_\sigma^V$ by turning them into, respectively, an eraser and a duplicator of a ground type. Then, we show that the reduction steps in Figure 3.4(a) and the commuting conversions in Figure 3.4(b) can be simulated by the $\beta\eta$-reduction of the linear λ-calculus, as long as we restrict to terms of $\Lambda_{l,\downarrow}$ typable in LEM. Last, we discuss the complexity of the translation, and we prove that every term typable in LEM is mapped to a linear λ-term whose size is exponential in the one of the original term.

We start with the following preliminary lemma:

Lemma 33. *Let V be a value and σ a type of* LEM*:*

(1) if $M \in \Lambda_{l,\downarrow}$ is a term typable in LEM*:*

- *every subterm of M of the form $\texttt{discard}_\sigma\ P\ \texttt{in}\ Q$ is such that σ is closed and free form negative occurrences of $\forall$, and P an inhabitant of σ;*
- *every subterm of M of the form $\texttt{copy}^V_\sigma\ P\ \texttt{as}\ x_1, x_2\ \texttt{in}\ Q$ is such that σ is closed and free form negative occurrences of $\forall$, and both V and P are inhabitants of σ.*

(2) if σ' is σ in which every occurrence of $\downarrow$ has been removed, then V has type σ if and only if V has type σ';

Proof. Straightforward. □

The translation from LEM to IMLL_2 can be defined as follows:

Definition 30 (From LEM to IMLL_2)**.** The map $(\_)^\bullet : \mathsf{LEM} \longrightarrow \mathsf{IMLL}_2$ translates a derivation $\mathcal{D} \triangleleft \Gamma \vdash_{\mathsf{LEM}} M : \tau$ into a derivation $\mathcal{D}^\bullet \triangleleft \Gamma^\bullet \vdash_{\mathsf{IMLL}_2} M^\bullet : \tau^\bullet$:

- for all types $\sigma \in \Theta_\downarrow$, it is defined by:

$$\alpha^\bullet \triangleq \alpha$$
$$(\tau \multimap A)^\bullet \triangleq \tau^\bullet \multimap A^\bullet$$
$$(\forall \alpha.A)^\bullet \triangleq \forall \alpha.A^\bullet$$
$$(\downarrow\tau)^\bullet \triangleq \tau^\bullet$$

- for all contexts $\Gamma = x_1 : \sigma_1, \dots, x_n : \sigma_n$, we set $\Gamma^\bullet \triangleq x_1 : \sigma_1^\bullet, \dots, x_n : \sigma_n^\bullet$;
- for all typable terms $M \in \Lambda_{l,\downarrow}$, it is defined by:

$$x^\bullet \triangleq x$$
$$(\lambda x.P)^\bullet \triangleq \lambda x.P^\bullet$$
$$(PQ)^\bullet \triangleq P^\bullet Q^\bullet$$
$$(\texttt{discard}_\sigma\ P\ \texttt{in}\ Q)^\bullet \triangleq \texttt{let}\ \texttt{E}_{\sigma^\bullet}\ P^\bullet\ \texttt{be}\ \mathbf{I}\ \texttt{in}\ Q^\bullet$$
$$(\texttt{copy}^V_\sigma\ P\ \texttt{as}\ x_1, x_2\ \texttt{in}\ Q)^\bullet \triangleq \texttt{let}\ \texttt{D}^{V^\bullet}_{\sigma^\bullet}\ P^\bullet\ \texttt{be}\ x_1, x_2\ \texttt{in}\ Q^\bullet$$

  where Theorem 9 and Theorem 10 assure the existence of the erasure $\texttt{E}_{\sigma^\bullet}$ and the duplicator $\texttt{D}^{V^\bullet}_{\sigma^\bullet}$ of the type $\sigma^\bullet$ (with the notation as in Remark 4), because by Lemma 33.(1):

  – σ is closed and free from negative occurrences of $\forall$, so that $\sigma^\bullet$ is a ground type,

  – $V^\bullet = V$ is a value that inhabits σ, and hence $\sigma^\bullet$, by Lemma 33.(2);

- the definition of $(\_)^\bullet$ extends to any derivation $\mathcal{D} \triangleleft \Gamma \vdash M : \sigma$ of LEM in the obvious way, following the structure of $M^\bullet$. Figure 3.7 collects the most interesting cases.

$$\left(\dfrac{\begin{array}{c}\mathcal{D}\\ x_1 : \downarrow\sigma_1, \ldots, x_n : \downarrow\sigma_n \vdash M : \sigma\end{array}}{x_1 : \downarrow\sigma_1, \ldots, x_n : \downarrow\sigma_n \vdash M : \downarrow\sigma}\, p \right)^{\bullet} \triangleq \left(\begin{array}{c}\mathcal{D}\\ x_1 : \downarrow\sigma_1, \ldots, x_n : \downarrow\sigma_n \vdash M : \sigma\end{array} \right)^{\bullet}$$

$$\left(\dfrac{\begin{array}{c}\mathcal{D}\\ \Gamma, x : \sigma \vdash M : \tau\end{array}}{\Gamma, y : \downarrow\sigma \vdash M[y/x] : \tau}\, d \right)^{\bullet} \triangleq \dfrac{\dfrac{}{y : \sigma^{\bullet} \vdash y : \sigma^{\bullet}}\,\mathrm{ax} \qquad \left(\begin{array}{c}\mathcal{D}\\ \Gamma, x : \sigma \vdash M : \tau\end{array} \right)^{\bullet}}{\Gamma^{\bullet}, y : \sigma^{\bullet} \vdash M^{\bullet}[y/x] : \tau^{\bullet}}\, \mathit{cut}$$

$$\left(\dfrac{\begin{array}{c}\mathcal{D}\\ \Gamma \vdash M : \tau\end{array}}{\Gamma, x : \downarrow\sigma \vdash \mathtt{discard}_{\sigma}\, x\ \mathtt{in}\ M : \tau}\, w \right)^{\bullet} \triangleq \dfrac{\begin{array}{c}\vdots\\ x : \sigma^{\bullet} \vdash \mathtt{E}_{\sigma^{\bullet}}\, x : \mathbf{1}\end{array} \qquad \dfrac{\left(\begin{array}{c}\mathcal{D}\\ \Gamma \vdash M : \tau\end{array} \right)^{\bullet}}{\Gamma^{\bullet}, y : \mathbf{1} \vdash \mathtt{let}\ y\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ M^{\bullet} : \tau^{\bullet}}\, \mathbf{1}\mathrm{L}}{\Gamma^{\bullet}, x : \sigma^{\bullet} \vdash \mathtt{let}\ \mathtt{E}_{\sigma^{\bullet}}\, x\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ M^{\bullet} : \tau^{\bullet}}\, \mathit{cut}$$

$$\left(\dfrac{\begin{array}{c}\mathcal{D}_1\\ \Gamma, x_1 : \downarrow\sigma, x_2 : \downarrow\sigma \vdash M : \tau\end{array} \qquad \begin{array}{c}\mathcal{D}_2\\ \vdash V : \sigma\end{array}}{\Gamma, x : \downarrow\sigma \vdash \mathtt{copy}^{V}_{\sigma}\, x\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ M : \tau}\, c \right)^{\bullet} \triangleq \dfrac{\begin{array}{c}\left(\begin{array}{c}\mathcal{D}_2\\ \vdash V : \sigma\end{array} \right)^{\bullet}\\ \vdots\\ x : \sigma^{\bullet} \vdash \mathtt{D}^{V^{\bullet}}_{\sigma^{\bullet}}\, x : \sigma^{\bullet} \otimes \sigma^{\bullet}\end{array} \qquad \dfrac{\left(\begin{array}{c}\mathcal{D}_1\\ \Gamma, x_1 : \downarrow\sigma, x_2 : \downarrow\sigma \vdash M : \tau\end{array} \right)^{\bullet}}{\Gamma^{\bullet}, y : \sigma^{\bullet} \otimes \sigma^{\bullet} \vdash \mathtt{let}\ y\ \mathtt{be}\ x_1, x_2\ \mathtt{in}\ M^{\bullet} : \tau^{\bullet}}\, \otimes\mathrm{L}}{\Gamma^{\bullet}, x : \sigma^{\bullet} \vdash \mathtt{let}\ \mathtt{D}^{V^{\bullet}}_{\sigma^{\bullet}}\, x\ \mathtt{be}\ x_1, x_2\ \mathtt{in}\ M^{\bullet} : \tau^{\bullet}}\, \mathit{cut}$$

Figure 3.7: The translation of the rules p, d, w and c.

Before stating the simulation theorem we prove a substitution lemma:

Lemma 34. *For all terms $M, N \in \Lambda_{l,\downarrow}$ typable in* LEM*, $M^\bullet[N^\bullet/x] = (M[N/x])^\bullet$.*

Proof. The proof is by structural induction on M. If M is x, then $x^\bullet[N^\bullet/x] = x[N^\bullet/x] = N^\bullet = (x[N/x])^\bullet$. If M is $\lambda y.P$ then, by using the induction hypothesis:

$$\begin{aligned}(\lambda y.P)^\bullet[N^\bullet/x] &= (\lambda y.P^\bullet)[N^\bullet/x] = \lambda y.(P^\bullet[N^\bullet/x]) = \lambda y.(P[N/x])^\bullet \\ &= (\lambda y.(P[N/x]))^\bullet = ((\lambda y.P)[N/x])^\bullet.\end{aligned}$$

If M is PQ, then either $x \in \mathrm{FV}(P)$ or $x \in \mathrm{FV}(Q)$. We consider the former case, the latter being similar. By using the induction hypothesis: $(PQ)^\bullet[N^\bullet/x] = P^\bullet[N^\bullet/x]Q^\bullet = (P[N/x])^\bullet Q^\bullet = (P[N/x]Q)^\bullet = (PQ)[N/x]^\bullet$. If M is $\mathtt{discard}_\sigma\, P\ \mathtt{in}\ Q$, then either $x \in P$ or $x \in Q$. We consider the former case. By using the induction hypothesis:

$$\begin{aligned}(\mathtt{discard}_\sigma\, P\ \mathtt{in}\ Q)^\bullet[N^\bullet/x] &= \mathtt{let}\ \mathtt{E}_{\sigma^\bullet}\ P^\bullet[N^\bullet/x]\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ Q^\bullet \\ &= \mathtt{let}\ \mathtt{E}_{\sigma^\bullet}\ (P[N/x])^\bullet\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ Q^\bullet \\ &= (\mathtt{discard}_\sigma\, P[N/x]\ \mathtt{in}\ Q)^\bullet.\end{aligned}$$

If M is $\mathtt{copy}^V_\sigma\, P\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ Q$ then either $x \in \mathrm{FV}(P)$ or $x \in \mathrm{FV}(Q)$. We consider the former case. By using the induction hypothesis:

$$\begin{aligned}(\mathtt{copy}^V_\sigma\, P\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ Q)^\bullet[N^\bullet/x] &= \mathtt{let}\ \mathtt{D}^{V^\bullet}_{\sigma^\bullet}\ P^\bullet[N^\bullet/x]\ \mathtt{be}\ x_1, x_2\ \mathtt{in}\ Q^\bullet \\ &= \mathtt{let}\ \mathtt{D}^{V^\bullet}_{\sigma^\bullet}\ (P[N/x])^\bullet\ \mathtt{be}\ x_1, x_2\ \mathtt{in}\ Q^\bullet \\ &= (\mathtt{copy}^V_\sigma\, P[N/x]\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ Q)^\bullet.\end{aligned}$$

□

We now show that every reduction on terms typable in LEM can be simulated in the linear λ-calculus by means of the $\beta\eta$-reduction. Since by Theorem 3 every linear λ-term has type in IMLL, and hence in IMLL_2, this result can be seen as a simulation property relating LEM and IMLL_2.

Theorem 35 (Simulation for LEM)**.** *Let $\mathcal{D} \lhd \Gamma \vdash M : \sigma$ be a derivation in* LEM*. If $M_1 \rightarrow^* M_2$ then $M_1^\bullet \rightarrow^*_{\beta\eta} M_2^\bullet$:*

$$\begin{tikzcd} M_1 \arrow[r, "*"] \arrow[d, dashed] & M_2 \arrow[d, dashed] \\ M_1^\bullet \arrow[r, "*", "\beta\eta"'] & M_2^\bullet \end{tikzcd}$$

Proof. By Theorem 32, it suffices to show that $M_1 \rightarrow M_2$ implies $M_1^\bullet \rightarrow^*_{\beta\eta} M_2^\bullet$. We proceed by case analysis and we consider the most interesting cases. Suppose that M_1 is $(\lambda x.P)Q$ and $M_2 = P[Q/x]$. Lemma 34 implies $((\lambda y.P)Q)^\bullet = (\lambda y.P^\bullet)Q^\bullet \rightarrow_\beta P^\bullet[Q^\bullet/x] = (P[Q/x])^\bullet$. If M_1 is $\mathtt{discard}_\sigma\, V\ \mathtt{in}\ N$ and M_2 is N, then V is a value of type σ by Lemma 33.(1), hence $V^\bullet = V$ is a value of type $\sigma^\bullet$ by Lemma 33.(2). Moreover, $\mathtt{E}_{\sigma^\bullet}$ is an eraser of $\sigma^\bullet$ by Definition 30. Therefore:

$$(\mathtt{discard}_\sigma\, V\ \mathtt{in}\ N)^\bullet \triangleq \mathtt{let}\ \mathtt{E}_{\sigma^\bullet}\ V^\bullet\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ N^\bullet \rightarrow^*_\beta N^\bullet$$

by Theorem 9. If M_1 is $\mathtt{copy}^{V'}_\sigma\, V\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ N$ and M_2 is $N[V/x_1, V/x_2]$, then V is a value of type σ by Lemma 33.(1), hence $V^\bullet = V$ is a value of type $\sigma^\bullet$ by Lemma 33.(2). Moreover, $\mathtt{D}^{(V')^\bullet}_{\sigma^\bullet}$ is a duplicator of $\sigma^\bullet$ by Definition 30. Therefore:

$$(\mathtt{copy}^{V'}_\sigma\, V\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ N)^\bullet \triangleq \mathtt{let}\ \mathtt{D}^{(V')^\bullet}_{\sigma^\bullet}\ V^\bullet\ \mathtt{be}\ x_1, x_2\ \mathtt{in}\ N^\bullet$$

$$\begin{aligned}
&\rightarrow^*_{\beta\eta} \texttt{let}\ \langle V^\bullet, V^\bullet\rangle\ \texttt{be}\ x_1, x_2\ \texttt{in}\ N^\bullet && \text{Thm. 10}\\
&\rightarrow_\beta N^\bullet[V^\bullet/x_1, V^\bullet/x_2]\\
&\triangleq (N^\bullet[V^\bullet/x_1])[V^\bullet/x_2]\\
&= ((N[V/x_1])[V/x_2])^\bullet && \text{Lem. 34}\\
&\triangleq (N[V/x_1, V/x_2])^\bullet.
\end{aligned}$$

□

We conclude by estimating the impact of the translation in Definition 30 on the size of terms, and hence the cost of "unpacking" the constructs $\texttt{discard}_\sigma$ and $\texttt{copy}^V_\sigma$. First, we study the complexity of erasers $\texttt{E}_A$ and duplicators $\texttt{D}^V_A$ with respect to A^-, where $(\_)^-$ is a forgetful map (erasing all instances of $\forall$ in a type) we introduced in Definition 12 (Section 3.2).

Lemma 36 (Size of $\texttt{E}_A$ and $\texttt{D}^V_A$)**.** *For every ground type A:*

(1) $|\texttt{E}_A| \in \mathcal{O}(|A^-|)$.

(2) If V is a value of type A, then $|\texttt{D}^V_A| \in \mathcal{O}(2^{|A^-|^2})$.

Proof. Point (1) is straightforward by looking at the proof of Theorem 9. Concerning point (2), from Section 3.2 we know that $\texttt{D}^V_A$ is $\texttt{dec}^s_A \circ \texttt{enc}^s_A \circ \texttt{sub}^V_A$, where $s = \mathcal{O}(|A^-| \cdot \log |A^-|)$. The components of $\texttt{D}^V_A$ with a size not linear in $|A^-|$ are $\texttt{dec}^s_A$ and $\texttt{enc}^s_A$. The λ-term $\texttt{dec}^s_A$ in Section 3.2.3 nests occurrences of $\texttt{if-then-else}$ each containing 2^s pairs of normal inhabitants of A, each one of size bounded by $|A^-|$, by Lemma 16. Similarly, $\texttt{enc}^s_A$ alternates instances of λ-terms $\texttt{abs}^s$ and $\texttt{app}^s$ which, again, nest occurrences of $\texttt{if-then-else}$ each one with 2^s instances of boolean strings of size s. Therefore, the overall complexity of $\texttt{D}^V_A$ is $\mathcal{O}(s \cdot 2^s) = \mathcal{O}(2^{|A^-|^2})$. □

Theorem 37 (Exponential compression for LEM)**.** *Let $\mathcal{D} \lhd \Gamma \vdash M : \sigma$ be a derivation in LEM. Then, $|M^\bullet| = \mathcal{O}(2^{|M|^k})$, for some $k \geq 1$.*

Proof. The proof is by structural induction on M. The only interesting case is when M is $\texttt{copy}^V_\sigma\ P\ \texttt{as}\ x_1, x_2\ \texttt{in}\ Q$, so that:

$$(\texttt{copy}^V_\sigma\ P\ \texttt{as}\ x_1, x_2\ \texttt{in}\ Q)^\bullet = \texttt{let}\ \texttt{D}^{V^\bullet}_{\sigma^\bullet}\ P^\bullet\ \texttt{be}\ x_1, x_2\ \texttt{in}\ Q^\bullet$$

where $\sigma^\bullet$ is a ground type of IMLL_2 with inhabitant $V^\bullet = V$, by Definition 30. By Lemma 16 it is safe to assume that V is a value with largest size among values of type $\sigma^\bullet$, so that V is a η-long normal form. On the one hand, by induction hypothesis, we obtain $|P^\bullet| = \mathcal{O}(2^{|P|^{k'}})$ and $|Q^\bullet| = \mathcal{O}(2^{|Q|^{k''}})$, for some $k', k'' \geq 1$. On the other hand, by applying both Lemma 36 and Lemma 16, we have $|\texttt{D}^V_{\sigma^\bullet}| = \mathcal{O}(2^{|(\sigma^\bullet)^-|^2}) = \mathcal{O}(2^{(2\cdot|V|)^2})$. Therefore, there exists $k \geq 1$ such that $|M^\bullet| = \mathcal{O}(2^{(|V|+|P|+|Q|+1)^k}) = \mathcal{O}(2^{|M|^k})$. □

3.4 The expressiveness of LEM and applications

Theorem 35 says that LEM is not "algorithmically" more expressive than IMLL_2. Nonetheless, terms with type in LEM, and their evaluations, exponentially compress the corresponding linear λ-terms and evaluations in IMLL_2 (Theorem 37). The goal of this section is to explore the benefits of this compression.

We first represent boolean circuits as terms of LEM, showing a simulation result (Proposition 38). The encoding is inspired by Mairson and Terui [65]. Other encodings of the boolean circuits have been proposed in Terui [89], Mogbil and Rahli [73] and Aubert [8] in the framework of the *unbounded* proof-nets for MLL, an efficient language able to express n-ary tensor products by single nodes and to characterize parallel computational complexity classes such as NC, AC, and $\mathsf{P}/_{\text{poly}}$. As opposed to all such previous approaches, our encoding exploits the use of `copy` and `discard` to directly express the fan-out nodes, allowing a more compact and modular representation of circuits. In particular, as compared to [89, 73, 8], our encoding is able to get rid of the garbage that accumulates in the course of the simulation.

Then, we present an encoding in LEM of the natural numbers similar to the standard Church encoding in Linear Logic, and we show that both the successor and the addition are definable (Proposition 40), while the "zero-test", the predecessor and the subtraction don't seem to be representable.

3.4.1 Boolean circuits in LEM

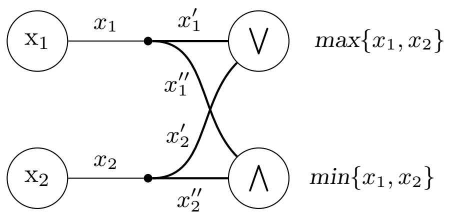
We start by briefly recalling the basics of boolean circuits from Vollmer [94].

Definition 31 (Boolean circuits)**.** A *boolean circuit* C is a finite, directed and acyclic graph with n *input nodes*, m *output nodes*, *internal nodes* and *fan-out nodes* as in Figure 3.8(a). The incoming (resp. outgoing) edges of a node are *premises* (resp. *conclusions*). The *fan-in* of an internal node is the number of its premises. Labels for the n input nodes of C are $\mathrm{x}_1, \dots, \mathrm{x}_n$ and those ones for the m outputs are $\mathrm{y}_1, \dots, \mathrm{y}_m$. Each internal node with fan-in $n \geq 0$ has an n-ary boolean function op^n as its label, provided that if $n = 0$, then op^n is a boolean constant in $\{0, 1\}$. The fan-out nodes have no label. Input and internal nodes are *logical nodes* and their conclusions are *logical edges*. If ν and ν' are logical nodes, then ν' is a *successor* of ν if a directed path from ν to ν' exists which crosses no logical node. The *size* $|C|$ of C is the number of nodes. Its *depth* $\delta(C)$ is the length of the longest path from an input node to a output node. A *basis* $\mathcal{B}$ is a set of boolean functions. A boolean circuit C is *over a basis* $\mathcal{B}$ if the label of every of its internal nodes belong to $\mathcal{B}$ only. The standard unbounded fan-in basis is $\mathcal{B}_1 = \{\neg, (\wedge^n)_{n\in\mathbb{N}}, (\vee^n)_{n\in\mathbb{N}}\}$.

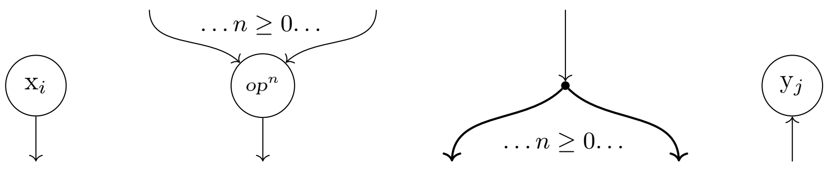
When representing boolean circuits as terms we label edges by λ-variables, we omit their orientation, we assume that every fan-out always has a logical edge as premise and we draw non-logical edges, i.e. conclusions of fan-out nodes, as thick lines.

Example 10. A 2-bits full-adder is in Figures 3.8(b). It takes two bits x_1, x_2 and a carrier c_{in} as inputs. Its outputs are the sum $s = (x_1 \oplus x_2) \oplus c_{in}$ and the carrier $c_{out} = (x_1 \wedge x_2) \vee ((x_1 \oplus x_2) \wedge c_{in})$, where $\oplus$ is the "exclusive or" that we can obtain by the functionally complete functions in $\mathcal{B}_1$.

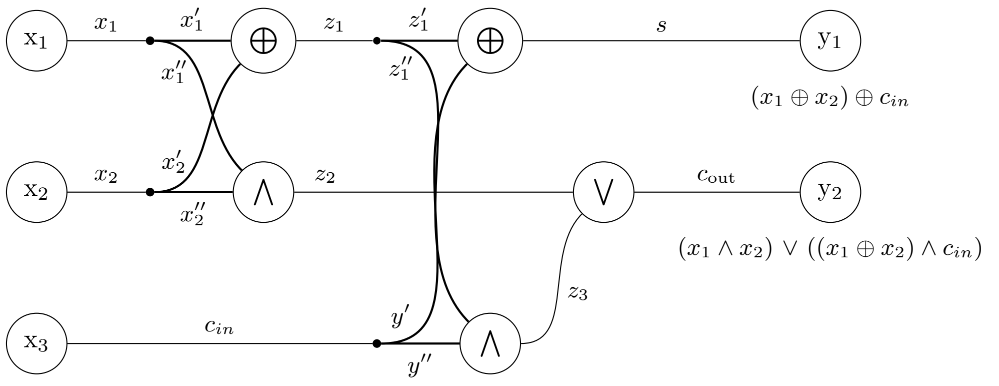
Example 11. The 3-bits majority function $maj_3(x_1, x_2, x_3)$ is in Figure 3.8(c). It serially composes three occurrences of the boolean circuit that switches two inputs x_1 and x_2 in order to put the greatest on the topmost output and the smallest on the bottommost one, under the convention that 0 is smaller than 1:



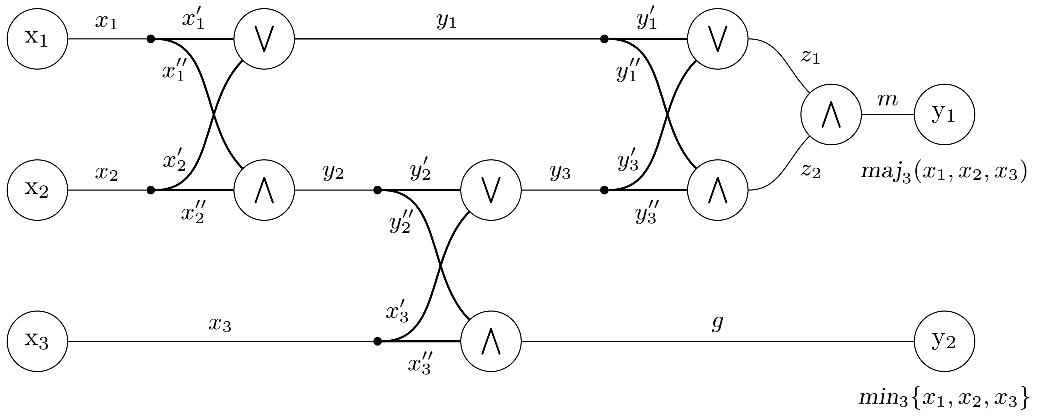
This kind of circuit is the building block of the Switching Networks [12]. So, the 3-bits majority circuit first sorts its input bits and then checks if the topmost two, i.e. the majority, are both set to 1. The lowermost output is garbage.



(a) From left, input, internal, fan-out, and output nodes.



(b) The 2-bits full-adder boolean circuit



(c) The 3-bits majority boolean circuit

Figure 3.8: Nodes of boolean circuits and some examples.

$\neg$	$\texttt{not} \triangleq \lambda b.\lambda x.\lambda y.byx$	$: \mathbf{B} \multimap \mathbf{B}$
$\wedge^0$	$\texttt{and}^0 \triangleq \lambda x.\lambda y.\langle x, y\rangle$	$: \mathbf{B}$
$\wedge^1$	$\texttt{and}^1 \triangleq \mathbf{I}$	$: \mathbf{B} \multimap \mathbf{B}$
$\wedge^2$	$\texttt{and}^2 \triangleq \lambda x_1.\lambda x_2.\pi_1(x_1 x_2\, \texttt{ff})$	$: \mathbf{B} \multimap \mathbf{B} \multimap \mathbf{B}$
$\wedge^{n+2}$	$\texttt{and}^{n+2} \triangleq \lambda x_1 \ldots x_{n+1} x_{n+2}.\texttt{and}^2\, (\texttt{and}^{n+1}\, x_1\, \ldots\, x_{n+1})\, x_{n+2}$	$: \mathbf{B} \multimap \overset{n+2}{\cdots} \multimap \mathbf{B} \multimap \mathbf{B}$
$\vee^0$	$\texttt{or}^0 \triangleq \lambda x.\lambda y.\langle y, x\rangle$	$: \mathbf{B}$
$\vee^1$	$\texttt{or}^1 \triangleq \mathbf{I}$	$: \mathbf{B} \multimap \mathbf{B}$
$\vee^2$	$\texttt{or}^2 \triangleq \lambda x_1.\lambda x_2.\pi_1(x_1 \texttt{tt}\, x_2)$	$: \mathbf{B} \multimap \mathbf{B} \multimap \mathbf{B}$
$\vee^{n+2}$	$\texttt{or}^{n+2} \triangleq \lambda x_1 \ldots x_{n+1} x_{n+2}.\texttt{or}^2\, (\texttt{or}^{n+1}\, x_1\, \ldots\, x_{n+1})\, x_{n+2}$	$: \mathbf{B} \multimap \overset{n+2}{\cdots} \multimap \mathbf{B} \multimap \mathbf{B}$
fo^0	$\texttt{out}^0 \triangleq \lambda x.\texttt{discard}_{\mathbf{B}}\, x\ \texttt{in}\ \mathbf{I}$	$: \downarrow\mathbf{B} \multimap \mathbf{1}$
fo^1	$\texttt{out}^1 \triangleq \mathbf{I}$	$: \downarrow\mathbf{B} \multimap \mathbf{B}$
fo^2	$\texttt{out}^2 \triangleq \lambda x.\texttt{copy}^{\texttt{tt}}_{\mathbf{B}}\, x\ \texttt{as}\ x_1, x_2\ \texttt{in}\ \langle x_1, x_2\rangle$	$: \downarrow\mathbf{B} \multimap \downarrow\mathbf{B} \otimes \downarrow\mathbf{B}$
fo^{n+2}	$\texttt{out}^{n+2} \triangleq \lambda x.\texttt{copy}^{\texttt{tt}}_{\mathbf{B}}\, x\ \texttt{as}\ x_1, x_2\ \texttt{in}\ \langle \texttt{out}^{n+1}\, x_1, x_2\rangle$	$: \downarrow\mathbf{B} \multimap \downarrow\mathbf{B} \otimes \overset{n+2}{\cdots} \otimes \downarrow\mathbf{B}$

Figure 3.9: Encoding of boolean functions and fan-out.

Translating boolean circuits as terms of LEM requires to encode the boolean functions in $\mathcal{B}_1$ and the fan-out nodes. Figure 3.9 reports them, where tt and ff encode the boolean values in (3.2), and π_1 is the projection in (3.5). By convention, we shall fix:

$$\underline{1} \triangleq \texttt{tt} \qquad\qquad \underline{0} \triangleq \texttt{ff}. \tag{3.23}$$

Moreover, $\texttt{op}^n$ denotes the n-ary boolean function op^n, according to Figure 3.9. We shorten $\texttt{and}^0$, $\texttt{or}^0$, $\texttt{and}^2$, $\texttt{or}^2$, and $\texttt{out}^2$ as $\underline{1}$, $\underline{0}$, and, or, and out, respectively. The encoding of the binary exclusive or $\oplus$ is xor.

We recall that boolean circuits are a model of parallel computation, while the λ-calculus models sequential computations. Mapping the former into the latter requires some technicalities. The notion of *level* allows to topologically sort the structure of the boolean circuits in order to preserve the node dependencies:

Definition 32 (Level)**.** The *level* l of a logical node ν in a boolean circuit C is:

- 0 if ν has no successors, and
- $\max\{l_1, \ldots, l_k\} + 1$ if ν has successors $\nu_1, \ldots, \nu_k$ with levels $l_1, \ldots, l_k$.

The *level* of a logical edge is the level of the logical node it is the conclusion of. The *level* of a boolean circuit is the greatest level of its logical nodes.

We define a level-by-level translation of unbounded fan-in boolean circuits over $\mathcal{B}_1$ into terms typable in LEM taking inspiration from Schubert [82]:

Definition 33 (From boolean circuits to terms)**.** Let C be a boolean circuit with n inputs and m outputs. We define the term $\texttt{level}^l_C$ by induction on $l - 1$:

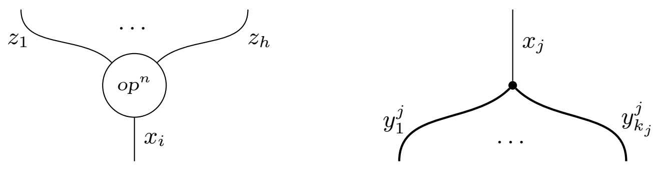


Figure 3.10: From left, an internal node and a fan-out node.

- $\texttt{level}^{-1}_C \triangleq \langle x_1, \dots, x_n \rangle$, where $x_1, \dots, x_n$ are the variables labelling the logical edges of level 0.
- $\texttt{level}^{l}_C \triangleq (\lambda x_1 \dots x_n x_{n+1} \dots x_m.\texttt{let}\,(\texttt{out}^{k_1}\, x_1)\ \texttt{be}\ y^1_1, \dots, y^1_{k_1}\ \texttt{in}\ \dots$ $\texttt{let}\,(\texttt{out}^{k_n}\, x_n)\ \texttt{be}\ y^n_1, \dots, y^n_{k_n}\ \texttt{in}\ \texttt{level}^{l-1}_C)\, B_1 \dots B_m$, where:
  - $x_1, \dots, x_n, x_{n+1}, \dots, x_m$ are the variables labelling the logical edges of level l;
  - for all $1 \leq j \leq n$, x_j is the premise of a fan-out node with conclusions labelled with $y^j_1, \dots, y^j_{k_j}$ (see Figure 3.10);
  - for all $1 \leq i \leq m$, if x_i is the variable labeling the conclusion of an internal node op^h with premises labeled by $z_1, \dots, z_h$, respectively (see Figure 3.10), then $B_i \triangleq \texttt{op}^h\, z_1\, \dots\, z_h$. If x_i is the variable labelling the conclusion of an input node then $B_i \triangleq x_i$.

Last, if the input nodes have conclusions labeled by $x_1, \dots, x_n$, respectively, and if C has level l, then we define $\lambda(C) \triangleq \lambda x.\texttt{let}\ x\ \texttt{be}\ x_1, \dots, x_n\ \texttt{in}\ \texttt{level}^l_C$.

Example 12 (2-bits full adder)**.** The level-by-level translation of the boolean circuit C in Figure 3.8(b) is the following:

$$\begin{aligned}
\texttt{level}^{-1}_C &\triangleq \langle s, c_{\text{out}} \rangle \\
\texttt{level}^{0}_C &\triangleq (\lambda s.\lambda c_{\text{out}}.\texttt{level}^{-1}_C)(\texttt{xor}\, z'_1\, y')(\texttt{or}\, z_2\, z_3) \\
\texttt{level}^{1}_C &\triangleq (\lambda z_2.\lambda z_3.\texttt{level}^{0}_C)(\texttt{and}\, x''_1\, x''_2)(\texttt{and}\, z''_1\, y'') \\
\texttt{level}^{2}_C &\triangleq (\lambda z_1.\lambda c_{\text{in}}.\texttt{let}\,(\texttt{out}\, z_1)\ \texttt{be}\ z'_1, z''_1\ \texttt{in} \\
&\qquad\qquad \texttt{let}\,(\texttt{out}\, c_{\text{in}})\ \texttt{be}\ y', y''\ \texttt{in}\ \texttt{level}^{1}_C)(\texttt{xor}\, x'_1\, x'_2)\, c_{\text{in}} \\
\texttt{level}^{3}_C &\triangleq (\lambda x_1.\lambda x_2.\texttt{let}\,(\texttt{out}\, x_1)\ \texttt{be}\ x'_1, x''_1\ \texttt{in} \\
&\qquad\qquad \texttt{let}\,(\texttt{out}\, x_2)\ \texttt{be}\ x'_2, x''_2\ \texttt{in}\ \texttt{level}^{2}_C)\, x_1\, x_2
\end{aligned}$$

where we set $\lambda(C) \triangleq \lambda x.\texttt{let}\, x\ \texttt{be}\ x_1, x_2, c_{\text{in}}\ \texttt{in}\ \texttt{level}^3_C$, that reduces to:

$$\begin{aligned}
&\lambda x.\texttt{let}\, x\ \texttt{be}\ x_1, x_2, c_{\text{in}}\ \texttt{in}\ (\texttt{let}\,(\texttt{out}\, x_1)\ \texttt{be}\ x'_1, x''_1\ \texttt{in} \\
&\texttt{let}\,(\texttt{out}\, x_2)\ \texttt{be}\ x'_2, x''_2\ \texttt{in}\ (\texttt{let}\,(\texttt{out}\, c_{\text{in}})\ \texttt{be}\ y', y''\ \texttt{in} \\
&\texttt{let}\,(\texttt{out}\,(\texttt{xor}\, x'_1\, x'_2))\ \texttt{be}\ z'_1, z''_2\ \texttt{in}\ \langle \texttt{xor}\, z'_1\, y', \texttt{or}\,(\texttt{and}\, x''_1\, x''_2)(\texttt{and}\, z''_1\, y'') \rangle\ )).
\end{aligned}$$

Example 13 (3-bits majority)**.** The level-by-level translation of the boolean circuit C in Figure 3.8(c) is the following:

$$
\begin{aligned}
\texttt{level}_C^{-1} &\triangleq \langle m, g\rangle\\
\texttt{level}_C^{0} &\triangleq (\lambda m.\lambda g.\texttt{level}_C^{-1})(\texttt{and}\, z_1\, z_2)(\texttt{and}\, y_2''\, x_3'')\\
\texttt{level}_C^{1} &\triangleq (\lambda z_1.\lambda z_2.\texttt{level}_C^{0})(\texttt{or}\, y_1'\, y_3')(\texttt{and}\, y_1''\, y_3'')\\
\texttt{level}_C^{2} &\triangleq (\lambda y_1.\lambda y_3.\texttt{let}\,(\texttt{out}\, y_1)\ \texttt{be}\ y_1', y_1''\ \texttt{in}\\
&\qquad\quad \texttt{let}\,(\texttt{out}\, y_3)\ \texttt{be}\ y_3', y_3''\ \texttt{in}\ \texttt{level}_C^{1})(\texttt{or}\, x_1'\, x_2')(\texttt{or}\, y_2'\, x_3')\\
\texttt{level}_C^{3} &\triangleq (\lambda y_2.\lambda x_3.\texttt{let}\,(\texttt{out}\, y_2)\ \texttt{be}\ y_2', y_2''\ \texttt{in}\\
&\qquad\quad \texttt{let}(\texttt{out}\, x_3)\ \texttt{be}\ x_3', x_3''\ \texttt{in}\ \texttt{level}_C^{2})(\texttt{and}\, x_1''\, x_2'')\, x_3\\
\texttt{level}_C^{4} &\triangleq (\lambda x_1.\lambda x_2.\texttt{let}\,(\texttt{out}\, x_1)\ \texttt{be}\ x_1', x_1''\ \texttt{in}\\
&\qquad\quad \texttt{let}\,(\texttt{out}\, x_2)\ \texttt{be}\ x_2', x_2''\ \texttt{in}\ \texttt{level}_C^{3})\, x_1\, x_2
\end{aligned}
$$

where we set $\lambda(C) \triangleq \lambda x.\texttt{let}\ x\ \texttt{be}\ x_1, x_2, x_3\ \texttt{in}\ \texttt{level}_C^4$, that reduces to:

$$
\begin{aligned}
&\lambda x.\texttt{let}\ x\ \texttt{be}\ x_1, x_2, x_3\ \texttt{in}\ \texttt{let}\,(\texttt{out}\, x_1)\ \texttt{be}\ x_1', x_1''\ \texttt{in}\\
&\texttt{let}\,(\texttt{out}\, x_2)\ \texttt{be}\ x_2', x_2''\ \texttt{in}\ (\texttt{let}\,(\texttt{out}\, x_3)\ \texttt{be}\ x_3', x_3''\ \texttt{in}\\
&\texttt{let}\,(\texttt{out}\,(\texttt{or}\, x_1'\, x_2'))\ \texttt{be}\ y_1', y_1''\ \texttt{in}\ (\texttt{let}\,(\texttt{out}\,(\texttt{and}\, x_1''\, x_2''))\ \texttt{be}\ y_2', y_2''\ \texttt{in}\\
&\texttt{let}\,(\texttt{out}\,(\texttt{or}\, y_2'\, x_3'))\ \texttt{be}\ y_3', y_3''\ \texttt{in}\ \langle \texttt{and}\,(\texttt{or}\, y_1'\, y_3')(\texttt{and}\ y_1''\, y_3''), \texttt{and}\, y_2''\, x_3''\rangle\ )).
\end{aligned}
$$

The size of the term coding an internal node depends on its fan-in. Likewise, the size of the term coding a fan-out node depends on the number of conclusions. The size of the circuit bounds both values. Moreover, by Theorem 32, reducing a typable term yields a typable term. These observations imply:

Proposition 38 (Simulation of circuit evaluation)**.** *Let C be an unbounded fan-in boolean circuit over $\mathcal{B}_1$ with n inputs and m outputs. Then:*

- *$\lambda(C)$ has type $(\downarrow\mathbf{B} \otimes \overset{n}{\ldots} \otimes \downarrow\mathbf{B}) \multimap (\mathbf{B} \otimes \overset{m}{\ldots} \otimes \mathbf{B})$ in* LEM*;*
- *$|\lambda(C)| = O(|C|)$;*
- *for all $(i_1, \ldots, i_n) \in \{0,1\}^n$, the evaluation of C on input $(i_1, \ldots, i_n)$ outputs the tuple $(i_1', \ldots, i_m') \in \{0,1\}^m$ if and only if $\lambda(C)\,\langle \underline{i}_1, \ldots, \underline{i}_n\rangle \to^* \langle \underline{i}_1', \ldots, \underline{i}_m'\rangle$.*

It should not be surprising that the translation cannot preserve the depth of a given circuit. Indeed, LEM has only *binary* logical operators, and we are forced to use nested instances of the construct "`let`" to access each A_i in the type $A_1 \otimes \ldots \otimes A_n$. We could preserve the depth by extending LEM with unbounded tensor products as done, for example, in [89] for the multiplicative fragment of linear logic MLL.

3.4.2 Numerals in LEM

We present in LEM an encoding of natural numbers quite close to the standard Church encoding.

Definition 34 ($\downarrow$-numerals)**.** The $\downarrow$-*numerals* can be defined as the following terms in $\Lambda_{l,\downarrow}$:

$$
\begin{aligned}
\overline{0} &\triangleq \lambda f x.\texttt{discard}_{\mathbf{1}}\, f\, \texttt{in}\, x\\
\overline{1} &\triangleq \lambda f x.f x \qquad\qquad (3.24)\\
\overline{n+2} &\triangleq \lambda f x.\texttt{copy}_{\mathbf{1}}^{\mathbf{I}}\, f\, \texttt{as}\, f_1 \ldots f_n\, \texttt{in}\, f_1(\ldots(f_n\, x)\ldots)
\end{aligned}
$$

where $\mathtt{copy}^{\mathbf{I}}_{\mathbf{1}}\, f_0 \,\mathtt{as}\, f_1 \ldots f_n \,\mathtt{in}\, M$ stands for:

$$\mathtt{copy}^{\mathbf{I}}_{\mathbf{1}}\, f_0 \,\mathtt{as}\, f_1, f'_2 \,\mathtt{in}\, (\mathtt{copy}^{\mathbf{I}}_{\mathbf{1}}\, f'_2 \,\mathtt{as}\, f_2, f'_3 \,\mathtt{in}\, \ldots (\mathtt{copy}^{\mathbf{I}}_{\mathbf{1}}\, f'_{n-1} \,\mathtt{as}\, f_{n-1}, f_n \,\mathtt{in}\, M) \ldots).$$

We define $\mathbf{int}_{\downarrow} \triangleq {\downarrow}\mathbf{1} \multimap \mathbf{1}$ as the type of $\downarrow$-numerals.

The following statement holds:

Proposition 39. *For all $n \in \mathbb{N}$, $\vdash_{\mathsf{LEM}} \overline{n} : \mathbf{int}_{\downarrow}$.*

In order to identify terms that represent the same natural number, we take $\downarrow$-numerals up to the following equivalences:

$$f(\mathtt{copy}^{\mathbf{I}}_{1}\, f' \,\mathtt{as}\, g, h \,\mathtt{in}\, M) \sim \mathtt{copy}^{\mathbf{I}}_{1}\, f' \,\mathtt{as}\, g, h \,\mathtt{in}\, f\, M. \tag{3.25}$$

$$\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, \mathtt{copy}^{\mathbf{I}}_{1}\, f_1 \,\mathtt{as}\, f_3, f_4 \,\mathtt{in}\, M \sim \mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_3, f_1 \,\mathtt{in}\, \mathtt{copy}^{\mathbf{I}}_{1}\, f_1 \,\mathtt{as}\, f_4, f_2 \,\mathtt{in}\, M. \tag{3.26}$$

Remark 7. Let us compare $\mathbf{int}_{\downarrow}$ with the type $\mathbf{int}$ of the Church numerals in Linear Logic:

$$\mathbf{int} \triangleq \forall\alpha.(!(\alpha \multimap \alpha) \multimap (\alpha \multimap \alpha)) \qquad \mathbf{int}_{\downarrow} \triangleq {\downarrow}(\forall\alpha.(\alpha \multimap \alpha)) \multimap \forall\alpha.(\alpha \multimap \alpha).$$

In the former the universal quantification is in positive position, while in the latter it occurs on both sides of the main implication, because the modality $\downarrow$ applies only to ground types, which are closed. We observe that the lack of an external quantifier in $\mathbf{int}_{\downarrow}$ limits the use of the $\downarrow$-numerals as iterators.

Recall that, by the Subject reduction property (Theorem 32), typability is preserved under reduction for terms in $\Lambda_{l,\downarrow}$. The following definition adapts to LEM the standard notion of representable function over $\mathbb{N}$ for type systems:

Definition 35 (Representable function)**.** We say that a function $f : \mathbb{N}^n \to \mathbb{N}$ is *representable in* LEM if there exists a term $F \in \Lambda_{l,\downarrow}$ with $\vdash F : \mathbf{int}_{\downarrow} \multimap \overset{n}{\ldots} \multimap \mathbf{int}_{\downarrow} \multimap \mathbf{int}_{\downarrow}$ such that:

$$F\overline{n_1} \ldots \overline{n_k} \to^* \overline{f(n_1, \ldots, n_k)}$$

for all $n_1, \ldots, n_k \in \mathbb{N}$.

We now show that the successor and the addition are representable in LEM. By pushing further the analogy with the Church encoding, we define:

$$\mathtt{S} \triangleq \lambda nfx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, f_1(nf_2x) : \mathbf{int}_{\downarrow} \multimap \mathbf{int}_{\downarrow}$$
$$\mathtt{A} \triangleq \lambda mnfx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, mf_1(nf_2x) : \mathbf{int}_{\downarrow} \multimap \mathbf{int}_{\downarrow} \multimap \mathbf{int}_{\downarrow}.$$

It is easy to check that:

$$\forall n, m \in \mathbb{N} \qquad \mathtt{S}\,\overline{n} \to^* \overline{n+1} \qquad \mathtt{A}\,\overline{m}\,\overline{n} \to^* \overline{m+n}.$$

We consider some examples here below:

Example 14. We show that $\mathtt{S}\,\overline{2} \to^* \overline{3}$:

$$\begin{aligned}
\mathtt{S}\,\overline{2} &\triangleq (\lambda nfx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, f_1(nf_2x))(\lambda gy.\mathtt{copy}^{\mathbf{I}}_{\mathbf{1}}\, g \,\mathtt{as}\, g_1, g_2 \,\mathtt{in}\, g_1(g_2\, y)) \\
&\to \lambda fx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, f_1((\lambda gy.\mathtt{copy}^{\mathbf{I}}_{\mathbf{1}}\, g \,\mathtt{as}\, g_1, g_2 \,\mathtt{in}\, g_1(g_2\, y))f_2x) \\
&\to \lambda fx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, f_1((\lambda y.\mathtt{copy}^{\mathbf{I}}_{\mathbf{1}}\, f_2 \,\mathtt{as}\, g_1, g_2 \,\mathtt{in}\, g_1(g_2\, y))x) \\
&\to \lambda fx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, f_1(\mathtt{copy}^{\mathbf{I}}_{\mathbf{1}}\, f_2 \,\mathtt{as}\, g_1, g_2 \,\mathtt{in}\, g_1(g_2\, x)) \\
&= \lambda fx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, (\mathtt{copy}^{\mathbf{I}}_{\mathbf{1}}\, f_2 \,\mathtt{as}\, g_1, g_2 \,\mathtt{in}\, f_1(g_1(g_2\, x))) \triangleq \overline{3}. && \text{by (3.25)}
\end{aligned}$$

Example 15. We show that $\mathtt{A}\,\overline{2}\,\overline{2} \to^* \overline{4}$:

$$
\begin{aligned}
\mathtt{A}\,\overline{2}\,\overline{2} &\triangleq (\lambda mnfx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, m f_1 (n f_2 x))\,\overline{2}\,\overline{2}\\
&\to (\lambda nfx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, \overline{2} f_1 (n f_2 x))\,\overline{2}\\
&\to \lambda fx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, \overline{2} f_1 (\overline{2} f_2 x)\\
&= \lambda fx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, \overline{2} f_1 ((\lambda gy.\mathtt{copy}^{\mathbf{I}}_{1}\, g \,\mathtt{as}\, g_3, g_4 \,\mathtt{in}\, g_3 (g_4\, y)) f_2 x)\\
&\to^* \lambda fx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\, \overline{2} f_1 (\mathtt{copy}^{\mathbf{I}}_{1}\, f_2 \,\mathtt{as}\, g_3, g_4 \,\mathtt{in}\, g_3 (g_4\, x))\\
&= \lambda fx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\\
&\quad ((\lambda gy.\mathtt{copy}^{\mathbf{I}}_{1}\, g \,\mathtt{as}\, g_1, g_2 \,\mathtt{in}\, g_1 (g_2\, y)) f_1 (\mathtt{copy}^{\mathbf{I}}_{1}\, f_2 \,\mathtt{as}\, g_3, g_4 \,\mathtt{in}\, g_3 (g_4\, x))\\
&\to^* \lambda fx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\\
&\quad (\mathtt{copy}^{\mathbf{I}}_{1}\, f_1 \,\mathtt{as}\, g_1, g_2 \,\mathtt{in}\, g_1 (g_2\, (\mathtt{copy}^{\mathbf{I}}_{1}\, f_2 \,\mathtt{as}\, g_3, g_4 \,\mathtt{in}\, g_3 (g_4\, x))))\\
&= \lambda fx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, f_1, f_2 \,\mathtt{in}\\
&\quad (\mathtt{copy}^{\mathbf{I}}_{1}\, f_1 \,\mathtt{as}\, g_1, g_2 \,\mathtt{in}\, (\mathtt{copy}^{\mathbf{I}}_{1}\, f_2 \,\mathtt{as}\, g_3, g_4 \,\mathtt{in}\, g_1 (g_2 (g_3 (g_4\, x)))) && \text{by (3.25)}\\
&= \lambda fx.\mathtt{copy}^{\mathbf{I}}_{1}\, f \,\mathtt{as}\, g_1, f_1 \,\mathtt{in}\\
&\quad (\mathtt{copy}^{\mathbf{I}}_{1}\, f_1 \,\mathtt{as}\, g_2, f_2 \,\mathtt{in}\, (\mathtt{copy}^{\mathbf{I}}_{1}\, f_2 \,\mathtt{as}\, g_3, g_4 \,\mathtt{in}\, g_1 (g_2 (g_3 (g_4\, x)))) \triangleq \overline{4}. && \text{by (3.26)}
\end{aligned}
$$

To sum up, we have:

Proposition 40. *The successor and the addition functions are representable in* LEM*.*

One can hardly prove that the "zero-test", the predecessor and the subtraction are representable in LEM. Consider for example the predecessor for **int** introduced by Roversi [78]:

$$
\begin{aligned}
\mathtt{P} &\triangleq \lambda nsz.n\, S[s]\, B[z] && \text{(Predecessor)}\\
S[M] &\triangleq \lambda p.\mathtt{let}\ p\ \mathtt{be}\ l, r\ \mathtt{in}\ \langle M, lr\rangle && \text{(Step function)}\\
B[N] &\triangleq \langle \mathbf{I}, N\rangle && \text{(Base function)}
\end{aligned}
$$

Giving a type to $\mathtt{P}$ requires to substitute $(\alpha \multimap \alpha)\otimes\alpha$ for α in **int**, as suggested by the application of $n : \mathbf{int}$ to $S[s]$. The position of the universal quantifiers in $\mathbf{int}_{\downarrow}$ forbids this operation, as already discussed in Remark 7. Otherwise, we could iterate functions, contradicting the cubic bound on the cut-elimination (Theorem 29).

To sum up, we have outlined the computational content of LEM, showing some number-theoretic functions that can be defined in the system and some others that cannot. However, we do not know of any characterization of the recursive functions representable in LEM. This is left to future investigations.

Chapter 4

Linear Additives and Probabilistic Polynomial Time

Soft Linear Logic (SLL) is a logic introduced by Lafont [56] to capture the complexity class P. It is a "light logic", i.e. a subsystem of Second-Order Linear Logic with weaker modal rules that limit the use of duplication, inducing a bound on normalization.

The Curry-Howard paradigm, stating a correspondence between formal logics and computational calculi, allows to see SLL as a type system, essentially by considering formulas as types and by giving a *complete* term decoration to logical derivations, as done in [9]. In this setting, a term is able to represent the whole structure of a derivation, so the bound on proof normalization can be turned into a bound on term evaluation.

This approach has a remarkable drawback: the presence of modal rules in SLL requires a heavy term annotation to faithfully encode derivations. A more reasonable strategy can be obtained by decorating proofs with terms from the standard λ-calculus. In this alternative setting, a λ-term does not fully represent a logical derivation, because it misses all the informations about the applications of modal rules. As a consequence, a bound on term evaluation is no longer inherited by the structural properties of derivations, and depends entirely on the typing conditions of SLL.

As pointed out by Gaboardi and Ronchi Della Rocca in [38, 40], the mismatch between λ-terms and logical derivations of SLL produced by the second approach prevents a close correspondence between proof normalization and term evaluation, leading to the failure of the Subject reduction property: in SLL typed terms exist that become untypable during evaluation. The failure of the Subject reduction has been first remarked by Lincoln [62] in the more general framework of Linear Logic, considered as a type assignment for the standard λ-calculus. To circumvent this problem, Gaboardi and Ronchi Della Rocca developed in [38, 40] *Soft Type Assignment* (STA), a subsystem of SLL obtained by restricting the set of types to the *essential ones*, that forbid modalities in the right-hand side of an implication, like $A \multimap !B$. STA enjoys the Subject reduction property and is expressive enough to capture the polynomial time.

In [39], Marion et al. investigate the system STA_+, an extension of STA characterizing the non-deterministic polynomial time. STA_+ is obtained from STA by endowing the λ-calculus with a non-deterministic choice operator "+", and by adding the *sum rule*, which derives the judgement $\Gamma \vdash M + N$ from the premises $\Gamma \vdash M : A$ and $\Gamma \vdash N : A$. In the resulting λ-calculus, a term of the form $M + N$ non-deterministically reduces either to M or to N. The sum rule of STA_+ is inspired by the namesake logical rule used by Maurel in [70] for the same purposes but in the context of LAL (Light Affine Logic), another light logic for the polynomial time (see [6, 7]).

Both in [39] and in [70], the Soundness theorem is achieved by defining a special evaluation strategy, as the presence of the sum rule produces normalizations that are exponential in time and space with respect to the size of the initial terms. For example, if we consider in STA_+ the typable term $(\lambda fx.f(\overset{n}{\ldots}(fx)))(\lambda y.zy + zy)\,v$ and we β-reduce it according to a innermost strategy, after $n+2$ steps we obtain a term with a number of redexes of the kind $M + N$ which is exponential in n.

The exponential blow up in normalization is a well-known drawback concerning all inference rules whose premises $\Gamma \vdash A$ share the same context Γ or the same predicate A. The additive rules in Linear Logic are a typical example. The inference rule &R introducing the additive connective & in the right-hand side of the turnstile allows to conclude $\Gamma \vdash \langle M, N\rangle : A \,\&\, B$ from the premises $\Gamma \vdash M : A$ and $\Gamma \vdash N : B$. The presence of &R gives rise to exponential proof normalizations, exactly as in STA_+ (see [65]). Nonetheless, Girard has shown that, by considering a special evaluation strategy called *lazy*, it is possible to recover a linear time normalization [46, 44].

The additive rules are closely related to non-determinism. Consequently, different non-deterministic formulations of these rules have been considered. In [67] Matsuoka extended Linear Logic with a self-dual additive connective whose cut-elimination involves choices. Moreover, the author has shown that several light logics are able to capture non-deterministic complexity classes when endowed with this new connective. A different approach is developed by Diaz-Caro [33]. The author shows that non-determinism arises naturally by replacing the usual construct $\pi_i : A_1 \& A_2 \multimap A_i$, projecting the i-th component of a pair, with the "ambiguous" construct $\pi_A : A \,\&\, A \multimap A$. Intuitively, the new operator π_A lacks the information about which component of a pair to project and requires a non-deterministic choice.

Recently, Horne [50] introduced restricted forms of additive connectives able to model probabilistic computation. The key observation that leads to these new connectives is that the standard additives cannot be faithfully interpreted as probabilistic processes, due to the presence of projections and injections. For example, if the pair $\langle \mathtt{head}, \mathtt{tail}\rangle$ of type $A \,\&\, B$ represents the process of "tossing a fair coin", the projection $\pi_1 : A \,\&\, B \multimap A$ (selecting the first component of $\langle \mathtt{head}, \mathtt{tail}\rangle$) would be forcefully interpreted as the process able to choose on which side to lay the coin, which is by no means probabilistic. Horne shows that, by ruling out both the projections and the injections from the standard additives, we obtain special connectives, the so-called *sub-additives*, in which a probabilistic interpretation can be recovered.

The results in [50] suggest that variants of the additives can be adopted to express probabilistic computations, and to capture probabilistic complexity classes in the style of light logics. To the best of our knowledge, however, this approach has not been pursued further.

The aim of this chapter is to fill this gap by exploiting the techniques of linear erasure and duplication we have investigated in Chapter 3. We first introduce *Linearly Additive Multiplicative Type Assignment* (LAM), a new type assignment system extending IMLL_2 with weaker formulations of the additive rules, called *linear additives*. This system enjoys a linear time normalization and the Subject reduction property. Moreover, as in the case of system LEM developed in the previous chapter, we define a translation of LAM into IMLL_2 that shows how linear additives relate to the mechanisms of linear weakening and contraction of IMLL_2.

What LAM witnesses is the role of the linear additive rules as costless formulations of the standard additives, with potentially fruitful applications to computational complexity. One of these applications is represented by $\mathsf{STA}_\oplus$, a new type system obtained by extending STA with a probabilistic variant of linear additives that is inspired by Diaz-Caro [33]. Linear additives provide the key ingredient to capture the probabilistic polynomial time. On the one hand, as opposed to the standard additives, their introduction does not affect the complexity of normalization. This allows $\mathsf{STA}_\oplus$ to naturally inherit the polynomial bound established for STA with no need of specific evaluation strategies to prevent the exponential blow up in computation. This is a step forward as compared to STA_+. On the other hand, linear additives are expressive enough to allow the complete encoding of a Probabilistic Turing Machine in $\mathsf{STA}_\oplus$.

Outline of the chapter. In this chapter we introduce the systems LAM and $\mathsf{STA}_\oplus$, and we prove that $\mathsf{STA}_\oplus$ characterizes the probabilistic polynomial time functions. In Section 4.1 we discuss the exponential blow up in normalization for IMALL_2 (Section 4.1.1) to motivate the introduction of the type system LAM (Section 4.1.2). We then explore some basic properties of LAM showing that this system enjoys the Subject reduction and a linear normalization (Section 4.1.3). We conclude the section by defining a translation of LAM into IMLL_2 (Section 4.1.4). In Section 4.2 we briefly recall SLL and STA (Section 4.2.1) in order to present the system $\mathsf{STA}_\oplus$ (Section 4.2.2), whose terms will be endowed with a probabilistic multi-step reduction. In Section 4.3 we establish a confluence result for the multi-step reduction (Section 4.3.1) and a "weighted" version of the Subject reduction property (Section 4.3.2) from which we infer the Soundness Theorem for $\mathsf{STA}_\oplus$ (Section 4.3.3). In Section 4.4 we describe the complete and detailed encoding in $\mathsf{STA}_\oplus$ of a Probabilistic Turing Machines that works in polynomial time (Sections 4.4.1 and 4.4.2), and we finally discuss the characterizations in $\mathsf{STA}_\oplus$ of the probabilistic complexity classes PP and BPP (Sections 4.4.3).

4.1 Linear additives

In this section we present the *Linearly Additive Multiplicative Type Assignment* (LAM), a new system extending IMLL_2 with a weaker formulation of the additive rules of Linear Logic called *linear additives* and based, quite like LEM in Chapter 3, on the mechanisms of linear contraction and weakening of IMLL_2. LAM is a type assignment for the term calculus $\Lambda_{l,\wedge}$ obtained by endowing Λ_l with type-depended constructs `copy` and `proj` that express, respectively, the sharing of variables in a pair and the projection of its components.

To motivate the introduction of LAM, and the need for a weaker formulation of the additives, we briefly recall IMALL_2 (Intuitionistic Second Order Multiplicative Additive Linear Logic) and we show an example of exponential normalization (Proposition 41).

Then we present LAM. The system enjoys the Subject reduction property (Theorem 46) and a linear normalization (Corollary 47). To conclude, as done in the previous chapter for LEM, we define a translation of LAM into IMLL_2 (Definition 43) and we prove a simulation result (Theorem 50), that shows how the construct `copy` exponentially compresses the mechanism of linear duplication of IMLL_2 (Theorem 51).

4.1.1 Toward linear additives: IMALL_2 and the exponential blow up

Second-Order Intuitionistic Multiplicative Additive Linear Logic (IMALL_2) is obtained by extending IMLL_2 with the so-called "additive rules" (see Section 2.2.2). We present this system as a type assignment for the λ-calculus Λ_π, endowed with explicit pairs and projections.

Definition 36 (The calculus Λ_π).

- Let $\mathcal{V}$ be a denumerable set of variables. The set Λ_π of *terms* is generated by the following grammar:
$$M := x \mid \lambda x.M \mid MM \mid (M,M) \mid \pi_1(M) \mid \pi_2(M) \tag{4.1}$$
where $x \in \mathcal{V}$.

- The set $FV(M)$ of free variables of a term M is standard for variables, abstractions and applications, and extends to the new clauses as follows:
$$\mathrm{FV}((M,N)) = \mathrm{FV}(M) \cup \mathrm{FV}(N)$$

$$\mathrm{FV}(\pi_i(M)) = \mathrm{FV}(M) \qquad\qquad i \in \{1,2\}.$$

The meta-level substitution of N for the free occurrences of x in M, written $M[N/x]$, is defined as usual. Similarly, the notion of size of a term M, written $|M|$, is extended to the new clauses as follows:

$$\begin{aligned} |(M,N)| &= |M| + |N| + 1 \\ |\pi_i(M)| &= |M| + 1 \qquad\qquad i \in \{1,2\}. \end{aligned}$$

A *context* is a term containing a unique *hole* $[\cdot]$, generated by the following grammar:

$$\mathcal{C} := \ [\cdot] \mid \lambda x.\mathcal{C} \mid \mathcal{C}M \mid M\mathcal{C} \mid (\mathcal{C}, M) \ \mid (M, \mathcal{C}) \mid \pi_1(\mathcal{C}) \mid \pi_2(\mathcal{C}) \tag{4.2}$$

where, given a context $\mathcal{C}$ and a term M, $\mathcal{C}[M]$ denotes the term obtained by subtituting the unique hole in $\mathcal{C}$ with M allowing the possible capture of free variables of M.

- The *one-step relation* $\rightarrow_{\beta\pi}$ is a binary relation over Λ_π. It is defined by the following rules:

$$\begin{aligned} (\lambda x.M)N &\rightarrow_{\beta\pi} M[N/x] \\ \pi_1(M_1, M_2) &\rightarrow_{\beta\pi} M_1 \\ \pi_2(M_1, M_2) &\rightarrow_{\beta\pi} M_2 \end{aligned} \tag{4.3}$$

  that apply in any context generated by (4.2). Its reflexive and transitive closure is $\rightarrow^*_{\beta\pi}$. As usual, a λ-term is in (or is a) *normal form* whenever no reduction rule applies to it.

We present IMALL_2 as a type assignment for Λ_π in natural deduction style.

Definition 37 (The system IMALL_2)**.**

- Let $\mathcal{X}$ be a denumerable set of type variables. The *types* of IMALL_2 are generated by the following grammar:

$$A := \ \alpha \mid A \multimap A \mid A \,\&\, A \mid \forall\alpha.A \tag{4.4}$$

  where $\alpha \in \mathcal{X}$. The standard meta-level substitution of B for every free-variables of α in A is denoted $A\langle B/\alpha\rangle$.

- IMALL_2 is the type assignment system for Λ_π displayed in Figure 4.1. It extends IMLL_2 in Figure 3.1(b) with the *additive rules* &I, &E1 and &E2.

Remark 8. The additives are closely related to non-determinism. Intuitively, the pair (M_1, M_2) introduced by the rule &I represents a sort of "stalemate" in computation: we cannot determine in advance which branching among M_1 and M_2 will be eventually chosen. The missing information to break this stalemate is then recovered by the rule &Ei, that introduces the projection π_i selecting the i-th component of a pair and discarding the other. As suggested in [67, 70, 33], one can give rise to non-deterministic computation by considering a variant of &Ei with a symmetric form of projection, let us denote it π, whose operational behaviour can be established only when interacting with a pair (M_1, M_2), i.e. when $\pi(M_1, M_2)$: in this case, either π behaves like π_1 and selects the first component of a pair, or it behaves like π_2 and selects the second component. More formally, the reduction rules corresponding to π would be the following ones:

$$\begin{aligned} \pi(M_1, M_2) &\rightarrow M_1 \\ \pi(M_1, M_2) &\rightarrow M_2 \end{aligned}$$

which introduce non-determinism in computation.

$$\frac{}{x : A \vdash x : A}\ ax$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B} \multimap\mathrm{I} \qquad \frac{\Gamma \vdash M : A \multimap B \qquad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B} \multimap\mathrm{E}$$

$$\frac{\Gamma \vdash M_1 : A_1 \qquad \Gamma \vdash M_2 : A_2}{\Gamma \vdash (M_1, M_2) : A_1 \,\&\, A_2} \&\mathrm{I} \qquad \frac{\Gamma \vdash M : A_1 \,\&\, A_2 \qquad i \in \{1, 2\}}{\Gamma \vdash \pi_i(M) : A_i} \&\mathrm{E}i$$

$$\frac{\Gamma \vdash M : A \qquad \gamma \notin \mathrm{FV}(\Gamma)}{\Gamma \vdash M : \forall\alpha.A} \forall\mathrm{I} \qquad \frac{\Gamma \vdash M : \forall\alpha.A}{\Gamma \vdash M : A\langle B/\alpha\rangle} \forall\mathrm{E}$$

Figure 4.1: The system IMALL_2.

Let us observe that the sharing of contexts in &I and the projection in &E, which play a fundamental role in the non-deterministic interpretation of Remark 8, express forms of contraction and weakening. Figure 4.2(b) provides an explicit and formal counterpart to this intuition by showing that the additive rules can be represented in the system IMLL_2 extended with the rules *contr* (*contraction*) and *weak* (*weakening*) of Figure 4.2(a), recalling that $\otimes$I and $\otimes$E are derivable in IMLL_2 (see Figure 3.2(c) and Definition 3).

The presence of implicit contractions in the additive rules affects the complexity of normalization by causing an exponential blow up, as we are going to show.

Definition 38 (Nesting IMALL_2 terms)**.** For all $n \in \mathbb{N}$ we define:

$$A_n \triangleq \begin{cases} A & \text{if } n = 0, \\ A_n \triangleq A_{n-1} \,\&\, A_{n-1} & \text{otherwise.} \end{cases}$$

For all $x \in \mathcal{V}$, for all $M \in \Lambda_\pi$, and for all $n \in \mathbb{N}$, we define:

$$\mathtt{add}^x_n \triangleq \begin{cases} x & \text{if } n = 0, \\ (\lambda y.\mathtt{add}^y_{n-1})(x, x) & \text{otherwise.} \end{cases} \qquad \mathtt{pair}^M_n \triangleq \begin{cases} M & \text{if } n = 0, \\ (\mathtt{pair}^M_{n-1}, \mathtt{pair}^M_{n-1}) & \text{otherwise.} \end{cases}$$

Observe that $\mathtt{pair}^M_n[N/x] = \mathtt{pair}^{M[N/x]}_n$. Moreover, notice that $\mathtt{pair}^{(x,x)}_n = \mathtt{pair}^x_{n+1}$, because $\mathtt{pair}^x_{n+1}$ has 2^{n+1} occurrences of x, and the term $\mathtt{pair}^{(x,x)}_n = \mathtt{pair}^x_n[(x, x)/x]$ doubles the number of occurrences of x in $\mathtt{pair}^x_n$, which is 2^n.

Proposition 41 (Exponential blow up)**.** *The following statements hold in* IMALL_2*:*

(1) for all $n, k \in \mathbb{N}$ both $\lambda x.\mathtt{add}^x_n$ and $\lambda x.\mathtt{pair}^x_n$ have type $A_k \multimap A_{k+n}$;

(2) for all $n \in \mathbb{N}$, $|\mathtt{add}^x_n| = \mathcal{O}(n)$ and $|\mathtt{pair}^x_n| = \mathcal{O}(2^n)$;

(3) for all $n \in \mathbb{N}$, $\mathtt{add}^x_n$ reduces to $\mathtt{pair}^x_n$ in n steps.

$$\frac{\Gamma, y : A, z : A \vdash M : B}{\Gamma, x : A \vdash M[x/y, x/z] : B}\,\mathit{contr} \qquad\qquad \frac{\Gamma \vdash M : B}{\Gamma, x : A \vdash M : B}\,\mathit{weak}$$

(a) The rules contraction and weakening

$$\frac{\dfrac{y_1 : A_1, \dots, y_n : A_n \vdash M_1 : B_1 \qquad z_1 : A_1, \dots, z_n : A_n \vdash M_2 : B_2}{y_1 : A_1, \dots, y_n : A_n, z_1 : A_1, \dots, z_n : A_n \vdash \langle M_1, M_2 \rangle : B_1 \otimes B_2}\,\otimes\mathrm{I}}{x_1 : A_1, \dots, x_n : A_n \vdash (M_1[x_1/y_1, \dots, x_n/y_n], M_2[x_1/z_1, \dots, x_n/z_n]) : B_1 \otimes B_2}\,\mathit{contr}$$

$$\frac{\Gamma \vdash M : A_1 \otimes A_2 \qquad \dfrac{\dfrac{}{y_i : A_i \vdash y_i : A_i}\,\mathit{ax}}{y_i : A_i, y_{3-i} : A_{3-i} \vdash y_i : A_i}\,\mathit{weak}}{\Gamma \vdash \mathtt{let}\ M\ \mathtt{be}\ y_i, y_{3-i}\ \mathtt{in}\ y_i : A_i}\,\otimes\mathrm{E}$$

(b) The additives rules are derivable in IMLL_2 extended with the rules *contr* and *weak*.

Figure 4.2: Additives hide contraction and weakening.

Proof. As for point (1), one can easily check that $\lambda x.\mathtt{pair}^x_n$ has type $A_k \multimap A_{k+n}$, for all $k \in \mathbb{N}$. We now prove by induction on n that $\lambda x.\mathtt{add}^x_n$ has type $A_k \multimap A_{k+n}$, for all $k \in \mathbb{N}$. If $n = 0$ then $\mathtt{add}^x_0 = x$, so that $\lambda x.x$ has type $A_k \multimap A_k$. Let us consider the case $n > 0$. By induction hypothesis, $\lambda y.\mathtt{add}^y_{n-1}$ has type $A_{k+1} \multimap A_{k+n}$. If x has type A_k then (x, x) has type A_{k+1} and $(\lambda y.\mathtt{add}^y_{n-1})(x, x)$ has type A_{k+n}. Therefore, $\lambda x.\mathtt{add}^x_n = \lambda x.((\lambda y.\mathtt{add}^y_{n-1})(x, x))$ has type $A_k \multimap A_{k+n}$. Concerning point (2), it suffices to prove by induction on n that $|\mathtt{add}^x_n| = (5 \cdot n) + 1$ and $|\mathtt{pair}^M_n| = 2^{n \cdot |M|} + \sum_{i=0}^{n-1} 2^i$ hold. Let us now prove point (3) by induction on n. The base case is trivial. If $n > 0$ then $\mathtt{add}^x_n = (\lambda y.\mathtt{add}^y_{n-1})(x, x)$ which reduces in one step to $\mathtt{add}^y_{n-1}[(x, x)/y]$. Since by induction hypothesis $\mathtt{add}^y_{n-1}$ reduces in $n - 1$ steps to $\mathtt{pair}^y_{n-1}$ then $\mathtt{add}^y_{n-1}[(x, x)/x]$ reduces in $n - 1$ steps to $\mathtt{pair}^y_{n-1}[(x, x)/y]$, which is $\mathtt{pair}^x_n$. $\square$

In other words, the nested term $\mathtt{add}^x_n$ reduces in n steps to $\mathtt{pair}^x_n$ whose size is $\mathcal{O}(2^n) = \mathcal{O}(2^{|\mathtt{add}_n|})$. Moreover, if M is a term with k redexes, then $\mathtt{add}^x_n[M/x]$ reduces to $\mathtt{pair}^x_n[M/x] = \mathtt{pair}^M_n$, and the number of redexes turns into $\mathcal{O}(2^k)$.

From the viewpoint of ICC, Proposition 41 seems to state that all type systems extending IMALL_2 are unable to capture complexity classes requiring at most a polynomial amount of time or space (e.g. PTIME, PSPACE, or NPTIME). This is not entirely true, as one can obtain a polynomial bound by considering specific reduction strategies. A typical example is *lazy reduction*, that evaluates a term of IMALL_2 both in linear time and in linear space [46], and allows to characterize PTIME in Light Linear Logic [44], a subsystem of Linear Logic with additive rules.

Lazy reduction "freezes" the evaluation inside a pair (M, N), so a term of IMLL_2 does not reduce to a normal form, in general. To recover a normalization result in IMALL_2 we need the notion of "lazy type":

Definition 39 (Lazy types for IMALL_2)**.** We say that A is a *lazy type* if it contains no negative occurrence of $\forall$ and no positive occurrence of &.

All inhabitants M of lazy type A always reach a normal form by lazy reduction, because each pair (P_1, P_2) in M eventually turns into a redex $\pi_i(P_1, P_2)$ that "unfreezes" the evaluation of P_i.

In the next subsection, we shall consider a radically different approach to circumvent the exponential blow up in normalization caused by the additives. We introduce LAM, a type system obtained by extending IMLL_2 with a weaker formulation of the additive rules, we shall call *linear additives*, that exploits the mechanisms of linear weakening and contraction discussed in the previous chapter.

4.1.2 The system LAM

The types of LAM are built from the linear implication "$\multimap$", the second-order quantification "$\forall$", and the new additive connective "$\wedge$", that applies only to closed types free from negative occurrences of $\forall$. As in the case of LEM in the previous chapter, these latter types will be the representatives in LAM of the ground types (Definition 10 of Section 3.1.2), and allow hidden forms of weakening and contraction in the new system.

Definition 40 (Types of LAM)**.** Let $\mathcal{X}$ be a denumerable set of type variables. The *types* of LAM are generated by the following grammar:

$$A, B := \ \alpha \mid A \multimap B \mid A \wedge B \mid \forall \alpha . A \tag{4.5}$$

where $\alpha \in \mathcal{X}$ and, in the clause $A \wedge B$ of (4.5), *both A and B must be closed and without negative occurrences of* $\forall$. The set of types generated by the grammar (4.5) will be denoted $\Theta_\wedge$. With $A\langle B/\alpha\rangle$ we denote the standard meta-level substitution of B for every occurrence of α in A. Finally, the *size* of a type A in $\Theta_\wedge$, written $|A|$, is the number of nodes in the syntax tree of A.

We shall define LAM as a type assignment for the term calculus $\Lambda_{l,\wedge}$, which is the standard linear λ-calculus endowed with a type-dependent construct $\mathtt{proj}_i^{A_1 \wedge A_2}$ for projection and a variant of $\mathtt{copy}_A^V$ from the term calculus $\Lambda_{l,\downarrow}$ (Definition 22), able to express the sharing of variables in a pair (M, N). These constructs are allowed to erase and duplicate what we call "extended value", a slightly more general notion of value (Definition 7).

Definition 41 (The calculi $\Lambda_\wedge$ and $\Lambda_{l,\wedge}$)**.**

- An *extended value* is any term in Λ_π that is either a closed and normal term of Λ_l or a pair (M, N), with M and N extended values. Extended values are ranged over by W.
- Let $\mathcal{V}$ be a denumerable set of variables. The set $\Lambda_\wedge$ of *terms* is generated by the following grammar:

$$M := \ x \mid \lambda x.M \mid MM \mid (M, M) \mid \mathtt{copy}_A^W\ M\ \mathtt{as}\ x, y\ \mathtt{in}\ (M, M) \mid \mathtt{proj}_i^{A_1 \wedge A_2}(M) \tag{4.6}$$

  where $x, y \in \mathcal{V}$, $i \in \{1, 2\}$, W is an extended value and $A, A_1, A_2 \in \Theta_\wedge$. We extend both $FV(M)$ and $|M|$ in Definition 36 to the new clauses:

$$\begin{aligned} \mathrm{FV}(\mathtt{copy}_A^W\ M\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (N_1, N_2)) &= \mathrm{FV}(M) \cup (\mathrm{FV}((N_1, N_2)) \setminus \{x_1, x_2\}) \\ \mathrm{FV}(\mathtt{proj}_i^{A_1 \wedge A_2}(M)) &= \mathrm{FV}(M) \qquad\qquad i \in \{1, 2\} \end{aligned}$$

$$|\mathtt{copy}_A^W\ M\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (N_1, N_2)| = |W| + |M| + |(N_1, N_2)| + 1$$

$$|\texttt{proj}_i^{A_1\wedge A_2}(M)| = |M| + 1 \qquad\qquad i \in \{1,2\}.$$

The meta-level substitution of N for the free occurrences of x in M, written $M[N/x]$, is defined as usual. A *context* is a term containing a unique *hole* $[\cdot]$ generated by the following grammar:

$$\begin{aligned}\mathcal{C} := [\cdot] \mid \lambda x.\mathcal{C} \mid \mathcal{C}M \mid M\mathcal{C} \mid (\mathcal{C}, M) \mid (M, \mathcal{C}) \mid \texttt{copy}_A^W\ \mathcal{C}\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (M_1, M_2)\\ \texttt{copy}_A^W\ M\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (\mathcal{C}, N) \mid \texttt{copy}_A^W\ M\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (N, \mathcal{C}) \mid \texttt{proj}_i^{A_1\wedge A_2}(\mathcal{C})\end{aligned} \tag{4.7}$$

where, given a context $\mathcal{C}$ and a term M, $\mathcal{C}[M]$ denotes the term obtained by subtituting the unique hole in $\mathcal{C}$ with M allowing the possible capture of free variables of M.

- The *one-step reduction relation* $\to$ is a binary relation over $\Lambda_\wedge$ defined by the following rules:

$$\begin{aligned}(\lambda x.M)N &\to M[N/x]\\ \texttt{proj}_1^{A_1\wedge A_2}((W_1, W_2)) &\to W_1\\ \texttt{proj}_2^{A_1\wedge A_2}((W_1, W_2)) &\to W_2\\ \texttt{copy}_A^{W'}\ W\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (M_1, M_2) &\to (M_1[W/x_1], M_2[W/x_2])\end{aligned} \tag{4.8}$$

  and can be applied in any context generated by (4.7), where W, W', W_1, W_2 are all extended values. Its reflexive and transitive closure is denoted $\to^*$. A term is in (or is a) *normal form* if no reduction applies to it.

- A term $M \in \Lambda_\wedge$ is *linear* if:
  - each free variable of M has just one free occurrence in it;
  - for each subterm $\lambda x.N$ of M, x occurs exactly once in N;
  - for each subterm (N_1, N_2) of M not in the scope of a $\texttt{copy}_A^W$ construct, $FV(N_1) = FV(N_2) = \emptyset$;
  - for each subterm $\texttt{copy}_A^{W'}\ P\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (N_1, N_2)$ of M, x_i occurs in N_i exactly once and $FV(N_1) = \{x_1\} \neq \{x_2\} = FV(N_2)$.

  The set of all linear terms in $\Lambda_\wedge$ is denoted $\Lambda_{l,\wedge}$.

Remark 9. A term $(N_1, N_2) \in \Lambda_\pi$ is typable in IMALL_2 just when $FV(N_1) = FV(N_2)$, i.e. when its components N_1 and N_2 *share* the same variables. This idea is maintained in $\Lambda_{l,\wedge}$, where a pair (N_1, N_2) can be of two kinds: either N_1 and N_2 are closed terms (representing a trivial form of sharing) or each N_i contains exactly one free variable x_i, and in this case (N_1, N_2) is endowed with a construct $\texttt{copy}$ expressing a linear form of sharing for the variables x_1 and x_2. So, for example, the term $\lambda xy.(x, y) \in \Lambda_\wedge$ is not a legal term of $\Lambda_{l,\wedge}$, while $\lambda z.\texttt{copy}_A^W\ z\ \texttt{as}\ x, y\ \texttt{in}\ (x, y)$ is. The latter can be seen as a linear counterpart of $\lambda z.(z, z) \in \Lambda_\pi$. Extended values are special terms in $\Lambda_{l,\wedge}$, so that $\lambda xy.(x, y)$ is not among them.

Proposition 42. *If $M \in \Lambda_{l,\wedge}$ and $M \to N$ then $N \in \Lambda_{l,\wedge}$.*

LAM is a type assignment system for $\Lambda_{l,\wedge}$ in natural deduction style. Its inference rules make meaningful both the type and the term annotations in the constructs $\texttt{proj}_1^{A_1\wedge A_2}$ and $\texttt{copy}_A^W$.

Definition 42 (The system LAM)**.** LAM is the type assignment system for $\Lambda_{l,\wedge}$ displayed in Figure 4.3, and extends IMLL_2 (Figure 3.1(b)) with the *linear additive rules* $\wedge$I1, $\wedge$I0, $\wedge$E1, and $\wedge$E2. It requires the following condition:

$$\frac{}{x : A \vdash x : A}\ ax$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B} \multimap\mathrm{I} \qquad \frac{\Gamma \vdash M : A \multimap B \qquad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B} \multimap\mathrm{E}$$

$$\frac{\vdash M_1 : A_1 \qquad \vdash M_2 : A_2}{\vdash (M_1, M_2) : A_1 \wedge A_2} \wedge\mathrm{I0} \qquad \frac{\Gamma \vdash M : A_1 \wedge A_2 \qquad i \in \{1,2\}}{\Gamma \vdash \mathtt{proj}_i^{A_1 \wedge A_2}(M) : A_i} \wedge\mathrm{E}i$$

$$\frac{\Gamma \vdash N : A \qquad x_1 : A \vdash M_1 : B_1 \qquad x_2 : A \vdash M_2 : B_2 \qquad \vdash W : A}{\Gamma \vdash \mathtt{copy}_A^W\ N\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (M_1, M_2) : B_1 \wedge B_2} \wedge\mathrm{I1}$$

$$\frac{\Gamma \vdash M : A \qquad \gamma \notin \mathrm{FV}(\Gamma)}{\Gamma \vdash M : \forall\alpha.A} \forall\mathrm{I} \qquad \frac{\Gamma \vdash M : \forall\alpha.A}{\Gamma \vdash M : A\langle B/\alpha\rangle} \forall\mathrm{E}$$

Figure 4.3: The system LAM.

- the type A in $\wedge$I1 *must be closed and free from negative occurrences of* $\forall$.

The linear additive rules $\wedge$I1, $\wedge$I0, $\wedge$E1, and $\wedge$E2 recall the additives in IMALL$_2$, and involve "hidden" applications of contraction and weakening similar to Figure 4.2. The crucial difference is that linear additives apply to types that are closed and free from negative occurrences of $\forall$, which are considered as the representatives in LAM of ground types (Definition 10). This ensures that each implicit contraction and weakening produced by the rules $\wedge$I1, $\wedge$I0, $\wedge$E1, and $\wedge$E2 can be faithfully mirrored by contractions and weakenings of ground types in IMLL$_2$. In particular, the rule $\wedge$I1 has two premises more than &I of IMALL$_2$. On the one hand, the premise with shape $\Gamma \vdash N : A$ is required to prove the Substitution property (Lemma 45). On the other hand, the premise with shape $\vdash W : A$ "witnesses" that A is inhabited by at least one (extended) value, quite like the rule c of LEM. This assures that the hidden contraction of the type A produced by the sharing of contexts represents a contraction of a ground type in IMLL$_2$, because Theorem 10 states that ground types are duplicable if *inhabited.*

All these intuitions will be formalized by defining a translation of LAM into IMLL$_2$ (Definition 43) showing how the constructs $\mathtt{copy}_A^W$ and $\mathtt{proj}_i^{A_1 \wedge A_2}$, that copy and discard extended values in $\Lambda_{l,\wedge}$, relate to duplicators and erasers of ground types (Theorem 50), that copy and discard values in the standard linear λ-calculus.

Finally, let us remark that the rules $\wedge$I1 and $\wedge$I0 represent two special cases of the rule &I of IMALL$_2$: the former has contexts with a single assumption, while the latter has no assumption at all (where the notation 1 and 0 in $\wedge$I1 and $\wedge$I0 is related to the number of assumptions of these rules). We adopt this presentation as just a matter of convenience: each "shared" assumption needs an explicit $\mathtt{copy}_A^W$ construct at the level of terms, and arbitrarily large contexts would produce a heavy notation. This has no real impact on the algorithmic expressiveness of the system, since a general inference rule $\wedge$In, with n assumptions on contexts, can be easily derived in the system. As an example, we show a derivation of the rule $\wedge$I2 in Figure 4.4(a), recalling that $\otimes$I and $\otimes$E are definable in IMLL_2, and hence in LAM.

We conclude by stating a property analogous to Lemma 16 for IMLL_2 and to Lemma 26.(2) for LEM:

Proposition 43. *Let W be and extended value and A be a closed type free from negative occurrences of $\forall$. If $\mathcal{D} \lhd \vdash W : A$ is a derivation of* LAM*, then $|W| \leq |A|$.*

Proof. It suffices to prove by an easy induction on the structure of terms that, for all normal terms $M \in \Lambda_{l,\wedge}$ such that $x_1 : A_1, \ldots, x_n : A_n \vdash M : A$, where A (resp. $A_1, \ldots, A_n$) is free from negative (resp. positive) occurrences of $\forall$, then $|M| \leq \sum_{i=1}^{n} |A_i| + |A|$. □

4.1.3 Subject reduction and linear normalization

Subject reduction requires some standard preliminary lemmas. With a little abuse of notation, in order to make the presentation easier we shall forget the distinction between free and bound type variables in the statement of the lemma below:

Lemma 44 (Generation)**.** *The following statements hold:*

(1) If $\mathcal{D} \lhd \Gamma \vdash x : A$, then $A = \forall \vec{\alpha}.(B\langle D_1/\beta_1, \ldots, D_n/\beta_n\rangle)$ and $\mathcal{D}$ is an instance of ax with conclusion $x : B \vdash x : B$ followed by a sequence of $\forall$I and $\forall$E, where $\vec{\alpha} = \alpha_1, \ldots, \alpha_m$, for some $m \geq 0$.

(2) If $\mathcal{D} \lhd \Gamma \vdash \lambda x.M : A$, then $A = \forall \vec{\alpha}.((B \multimap C)\langle D_1/\beta_1, \ldots, D_n/\beta_n\rangle)$ and $\mathcal{D}$ is some $\mathcal{D}' \lhd \Gamma, x : B \vdash M : C$ followed by $\multimap$ I and a sequence of $\forall$I and $\forall$E, where $\vec{\alpha} = \alpha_1, \ldots, \alpha_m$, for some $m \geq 0$.

(3) If $\mathcal{D} \lhd \Gamma \vdash MN : A$, then $A = \forall \vec{\alpha}.(C\langle D_1/\beta_1, \ldots, D_n/\beta_n\rangle)$ and $\mathcal{D}$ is some $\mathcal{D}' \lhd \Gamma' \vdash M : B \multimap C$ and $\mathcal{D}'' \lhd \Gamma'' \vdash N : B$ followed by $\multimap$E and a sequence of $\forall$I and $\forall$E, where $\Gamma = \Gamma', \Gamma''$ and $\vec{\alpha} = \alpha_1, \ldots, \alpha_m$, for some $m \geq 0$.

(4) If $\mathcal{D} \lhd \Gamma \vdash \mathtt{copy}_A^W\ N\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (M_1, M_2) : B$, then $B = C_1 \wedge C_2$ and the last rule of $\mathcal{D}$ is $\wedge$I1.

(5) If $\mathcal{D} \lhd \Gamma \vdash (M_1, M_2) : A$, then $\Gamma = \emptyset$, $A = B_1 \wedge B_2$ and the last rule of $\mathcal{D}$ is $\wedge$I0.

(6) If $\mathcal{D} \lhd \Gamma \vdash \mathtt{proj}_i^{B_1 \wedge B_2}(M) : A$ then $A = \forall \vec{\alpha}.(B_i'\langle D_1/\beta_1, \ldots, D_n/\beta_n\rangle)$ and $\mathcal{D}$ is some $\mathcal{D}' \lhd \Gamma \vdash M : B_1 \wedge B_2$ followed by $\wedge$Ei and a sequence of $\forall$I and $\forall$E, where $\vec{\alpha} = \alpha_1, \ldots, \alpha_m$, for some $m \geq 0$.

Proof. Straightforward, because the natural deduction system is essentially syntax directed. □

Lemma 45 (Linear substitution for LAM)**.** *Let $\mathcal{D}_1 \lhd \Gamma, x : A \vdash M : C$ and $\mathcal{D}_2 \lhd \Delta \vdash N : A$. Then there exists a derivation $S(\mathcal{D}_1, \mathcal{D}_2)$ such that:*

- *$S(\mathcal{D}_1, \mathcal{D}_2) \lhd \Gamma, \Delta \vdash M[N/x] : C$,*
- *$|M[N/x]| = |M| + |N| - 1$.*

Proof. The proof is by induction on $\mathcal{D}_1$. If the last rule is *ax* then $M = x$. We set $S(\mathcal{D}_1, \mathcal{D}_2) = \mathcal{D}_2$, so that $|M[N/x]| = |N|$. Suppose $\mathcal{D}_1$ is of the form:

$$\dfrac{\mathcal{D}' \lhd \Gamma, x : A \vdash P : B \quad \mathcal{D}'' \lhd x_1 : B \vdash Q_1 : C_1 \quad \mathcal{D}''' \lhd x_2 : B \vdash Q_2 : C_2 \quad \mathcal{D}'''' \lhd \vdash W : B}{\Gamma, x : A \vdash \mathtt{copy}_B^W\ P\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (Q_1, Q_2) : C_1 \wedge C_2}\ \wedge\text{I1}$$

$$\dfrac{\dfrac{\Gamma \vdash N_1 : A_1 \qquad \Gamma \vdash N_2 : A_2}{\Gamma, \Delta \vdash \langle N_1, N_2\rangle : A_1 \otimes A_2}\otimes\mathrm{I} \qquad \left\{\begin{array}{c} x_i : A_1, y_i : A_2 \vdash M_i : C_i \\ \vdots \\ z_i : A_1 \otimes A_2 \vdash \texttt{let}\ z_i\ \texttt{be}\ x_i, y_i\ \texttt{in}\ M_i : C_i \end{array}\right\}_{i\in\{1,2\}} \qquad \dfrac{\vdash W_1 : A_1 \qquad \vdash W_2 : A_2}{\vdash \langle W_1, W_2\rangle : A_1 \otimes A_2}\otimes\mathrm{I}}{\Gamma, \Delta \vdash \mathtt{copy}^{\langle W_1, W_2\rangle}_{A_1\otimes A_2}\ \langle N_1, N_2\rangle\ \texttt{as}\ z_1, z_2\ \texttt{in}\ (\texttt{let}\ z_1\ \texttt{be}\ x_1, y_1\ \texttt{in}\ M_1, \texttt{let}\ z_2\ \texttt{be}\ x_2, y_2\ \texttt{in}\ M_2) : C_1 \wedge C_2}\wedge\mathrm{I1}$$

(a) Derivation of $\wedge$I2.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z : \mathbf{B}_2 \vdash z : \mathbf{B}_2}}{\vdash \lambda z.z : \mathbf{B}_2 \multimap \mathbf{B}_2}\multimap\mathrm{I} \qquad \begin{array}{c}\vdots\\ y : \mathbf{B}_1 \vdash \mathtt{copy}^{(\mathtt{tt},\mathtt{tt})}_{\mathbf{B}_1}\ y\ \texttt{as}\ y_1, y_2\ \texttt{in}\ (y_1, y_2) : \mathbf{B}_2\end{array}}{y : \mathbf{B}_1 \vdash (\lambda z.z)(\mathtt{copy}^{(\mathtt{tt},\mathtt{tt})}_{\mathbf{B}_1}\ y\ \texttt{as}\ y_1, y_2\ \texttt{in}\ (y_1, y_2)) : \mathbf{B}_2}\multimap\mathrm{E}}{\vdash \lambda y.((\lambda z.z)(\mathtt{copy}^{(\mathtt{tt},\mathtt{tt})}_{\mathbf{B}_1}\ y\ \texttt{as}\ y_1, y_2\ \texttt{in}\ (y_1, y_2))) : \mathbf{B}_1 \multimap \mathbf{B}_2}\multimap\mathrm{I} \qquad \begin{array}{c}\vdots\\ x : \mathbf{B} \vdash \mathtt{copy}^{\mathtt{tt}}_{\mathbf{B}}\ x\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (x_1, x_2) : \mathbf{B}_1\end{array}}{x : \mathbf{B} \vdash (\lambda y.((\lambda z.z)(\mathtt{copy}^{(\mathtt{tt},\mathtt{tt})}_{\mathbf{B}_1}\ y\ \texttt{as}\ y_1, y_2\ \texttt{in}\ (y_1, y_2))))(\mathtt{copy}^{\mathtt{tt}}_{\mathbf{B}}\ x\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (x_1, x_2)) : \mathbf{B}_2}\multimap\mathrm{E}}{\vdash \lambda x.(\lambda y.((\lambda z.z)(\mathtt{copy}^{(\mathtt{tt},\mathtt{tt})}_{\mathbf{B}_1}\ y\ \texttt{as}\ y_1, y_2\ \texttt{in}\ (y_1, y_2))))(\mathtt{copy}^{\mathtt{tt}}_{\mathbf{B}}\ x\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (x_1, x_2)) : \mathbf{B} \multimap \mathbf{B}_2}\multimap\mathrm{I}$$

(b) Derivation in LAM of the term corresponding to $\lambda x.\mathtt{add}_2^x : \mathbf{B} \multimap \mathbf{B}_2$, where $\mathbf{B}_2 \triangleq \mathbf{B}_1 \wedge \mathbf{B}_1$ and $\mathbf{B}_1 \triangleq \mathbf{B} \wedge \mathbf{B}$.

Figure 4.4: Some derivations in LAM.

so that $C = C_1 \wedge C_2$ and $M = \texttt{copy}^W_B\ P\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (Q_1, Q_2)$. By induction hypothesis, there exists $S(\mathcal{D}', \mathcal{D}_2) \lhd \Gamma \vdash P[N/x] : A$ such that $|P[N/x]| = |P| + |N| - 1$. We define $S(\mathcal{D}_1, \mathcal{D}_2) \lhd \Gamma, \Delta \vdash \texttt{copy}^W_B\ P[N/x]\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (Q_1, Q_2) : C_1 \wedge C_2$ as the derivation obtained by applying $\wedge$I1 to $S(\mathcal{D}', \mathcal{D}_2)$, $\mathcal{D}''$, $\mathcal{D}'''$, $\mathcal{D}''''$. Moreover, by using the induction hypothesis, we have:

$$\begin{aligned} |M[N/x]| &= |W| + |P[N/x]| + |(Q_1, Q_2)| + 1 \\ &= |W| + |P| + |N| + |(Q_1, Q_2)| = |M| + |N| - 1. \end{aligned}$$

The other cases are similar. □

The Subject reduction property says that typability is preserved under reduction. We actually prove a stronger formulation of this property for LAM, stating also that the size of typable terms strictly decreases during reduction.

Theorem 46 (Subject reduction for LAM)**.** *If $\mathcal{D}_1 \lhd M_1 : A$ and $M_1 \to M_2$ then:*

- *there exists $\mathcal{D}_2$ such that $\mathcal{D}_2 \lhd \Gamma \vdash M_2 : A$;*
- $|M_2| < |M_1|$.

Proof. The proof proceeds by cases analysis on $M_1 \to M_2$. We just consider the most interesting cases:

- Suppose $M_1 = (\lambda x.P)Q$ and $M_2 = P[Q/x]$. By applying Lemma 44.(2) and (3), $\mathcal{D}$ must be as follows:

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}' \\ \Gamma', x : B \vdash P : C\end{matrix}}{\Gamma' \vdash \lambda x.P : B \multimap C}\multimap\text{I} \qquad \begin{matrix}\mathcal{D}'' \\ \Gamma'' \vdash Q : B\end{matrix}}{\Gamma', \Gamma'' \vdash (\lambda x.P)Q : C}\multimap\text{E}$$

$$\vdots\ \gamma$$

$$\Gamma \vdash (\lambda x.P)Q : A$$

where $\Gamma = \Gamma', \Gamma''$ and γ is a sequence of $\forall$I and $\forall$E. By applying Lemma 45 there exists a derivation $S(\mathcal{D}', \mathcal{D}'')$ such that $S(\mathcal{D}', \mathcal{D}'') \lhd \Gamma \vdash P[Q/x] : C$ and $|P[Q/x]| = |P| + |Q| - 1$. We define $\mathcal{D}_2$ as the following derivation:

$$\begin{matrix} S(\mathcal{D}', \mathcal{D}'') \\ \Gamma', \Gamma'' \vdash P[Q/x] : C \\ \vdots\ \gamma \\ \Gamma \vdash P[Q/x] : A \end{matrix}$$

where, $|M[N/x]| = |M| + |N| - 1 < |(\lambda x.M)N|$.

- Suppose $M_1 = \texttt{copy}^{W'}_A\ W\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (Q_1, Q_2)$ and $M_2 = (Q_1[W/x_1], Q_2[W/x_2])$. Then, by Lemma 44.(4), $A = C_1 \wedge C_2$ and $\mathcal{D}$ is as follows:

$$\dfrac{\mathcal{D}' \lhd \Gamma \vdash W : B \quad \mathcal{D}'' \lhd x_1 : B \vdash Q_1 : C_1 \quad \mathcal{D}''' \lhd x_2 : B \vdash Q_2 : C_2 \quad \mathcal{D}'''' \lhd \vdash W' : B}{\Gamma \vdash \texttt{copy}^{W'}_B\ W\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (Q_1, Q_2) : C_1 \wedge C_2}\wedge\text{I1}$$

We apply Lemma 45 twice and we get that there exist two derivations $S(\mathcal{D}', \mathcal{D}'') \triangleleft \vdash Q_1[W/x_1] : C_1$ and $S(\mathcal{D}', \mathcal{D}''') \triangleleft \vdash Q_2[W/x_2] : C_2$ such that $|Q_i[W/x_i]| = |Q_i| + |W| - 1$ for $i \in \{1, 2\}$. We define $\mathcal{D}_2$ as the following derivation:

$$\dfrac{\begin{array}{c} S(\mathcal{D}', \mathcal{D}'') \\ \vdash Q_1[W/x_1] : C_1 \end{array} \qquad \begin{array}{c} S(\mathcal{D}', \mathcal{D}''') \\ \vdash Q_2[W/x_2] : C_2 \end{array}}{\vdash (Q_1[W/x_1], Q_2[W/x_2]) : C_1 \wedge C_2} \wedge\mathrm{I0}$$

By Proposition 43 we can safely assume that W' has largest size among the extended values with type B. Therefore:

$$\begin{aligned} |M_2| &= |Q_1[W/x_1]| + |Q_2[W/x_2]| + 1 = 2 \cdot |W| + |Q_1| + |Q_2| - 1 \\ &< |W'| + |W| + |(Q_1, Q_2)| + 1 = |M_1|. \end{aligned}$$

- Suppose $M_1 = \texttt{proj}_i^{B_1 \wedge B_2}(W_1, W_2)$ and $M_2 = W_i$. By applying Lemma 44.(5) and Lemma 44.(6), $\Gamma = \emptyset$ and $\mathcal{D}$ must be as follows:

$$\begin{array}{c} \dfrac{\dfrac{\begin{array}{c} \mathcal{D}^1 \\ \vdash W_1 : B_1 \end{array} \qquad \begin{array}{c} \mathcal{D}^2 \\ \vdash W_2 : B_2 \end{array}}{\vdash (W_1, W_2) : B_1 \wedge B_2} \wedge\mathrm{I0}}{\vdash \texttt{proj}_i^{B_1 \wedge B_2}(W_1, W_2) : B_i} \wedge\mathrm{E}i \\ \vdots\, \gamma \\ \vdash \texttt{proj}_i^{B_1 \wedge B_2}(W_1, W_2) : A \end{array}$$

where γ is a sequence of $\forall$I and $\forall$E. We define $\mathcal{D}_2$ as follows:

$$\begin{array}{c} \mathcal{D}^i \\ \vdash W_i : B_i \\ \vdots\, \gamma \\ \vdash W_i : A \end{array}$$

where $|W_i| < |\texttt{proj}_i^{B_1 \wedge B_2}(W_1, W_2)|$.

$\square$

A straightforward corollary of Theorem 46 is the following:

Corollary 47 (Linear normalization for LAM)**.** *Let $\mathcal{D} \triangleleft \Gamma \vdash M : A$ be a lazy derivation. Then M reduces to a normal form in linear time.*

According to Corollary 47, no exponential blow up can be produced during normalization in LAM, as opposed to what happens in IMALL_2 (Proposition 41). Nonetheless, all IMALL_2 terms of the form $\lambda x.\texttt{add}_n^x : A_k \multimap A_{k+n}$ in Definition 38 are representable by suitable typable terms of LAM, provided that A is taken closed and free from negative occurrences of $\forall$. As an example, Figure 4.4(b) shows the encoding of $\lambda x.\texttt{add}_2^x : \mathbf{B} \multimap \mathbf{B}_2$ in LAM, where $\mathbf{B}$ is the type of booleans in (3.2), $\mathbf{B}_2 = \mathbf{B}_1 \,\&\, \mathbf{B}_1$, and $\mathbf{B}_1 = \mathbf{B} \,\&\, \mathbf{B}$.

How is this possible? The crucial point is in the way duplication works in LAM. As compared to IMALL_2, in which we can copy inhabitants of any type and with arbitrarily large size, in LAM duplication is restricted to closed types free from negative occurrences of $\forall$, and to extended values. On the one hand, since duplication applies only to normal forms, redexes cannot be copied during reduction, thus preventing an exponential time normalization. On the other hand, since by Proposition 43 these types have only *finitely many* extended values among their inhabitants, we can always predict and bound the increase of size produced by duplicating an extended value, thus preventing a size explosion during normalization.

4.1.4 Translation of LAM into IMLL_2 and exponential compression

In Section 3.4 a translation of LEM into IMLL_2 has been defined which shows how the constructs $\mathtt{discard}_\sigma$ and $\mathtt{copy}^V_\sigma$ relate to the mechanisms of linear erasure and duplication of IMLL_2. We follow essentially the same approach for LAM by mapping the constructs $\mathtt{proj}_i^{A_1 \wedge A_2}$ and $\mathtt{copy}^W_A$ to erasers and duplicators of ground types of IMLL_2.

We start with the following preliminary lemma:

Lemma 48. *Let W be an extended value and A be a type of* LAM*:*

(1) if $M \in \Lambda_{l,\wedge}$ is a term typable in LAM*:*

- *every subterm of M of the form $\mathtt{proj}_i^{A_1 \wedge A_2}(N)$ is such that A_1, A_2 are closed and free from negative occurrences of $\forall$, and P is an inhabitant of $A_1 \wedge A_2$;*
- *every subterm of M of the form $\mathtt{copy}^W_A\ P\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (Q_1, Q_2)$ is such that A is closed and free form negative occurrences of $\forall$, and both W, P are inhabitants of A;*

(2) if A' (resp. W') is A (resp. W) in which every subtype $B \wedge C$ (resp. every subterm (M, N)) occurring in it has been replaced by $B \otimes C$ (resp. $\langle M, N \rangle$), then W has type A if and only if W' has type A'.

Proof. Straightforward. □

The translation from LAM to IMLL_2 can be defined as follows:

Definition 43 (From LAM to IMLL_2)**.** The map $(\_)^\bullet : \mathsf{LAM} \longrightarrow \mathsf{IMLL}_2$ translates a derivation $\mathcal{D} \lhd \Gamma \vdash_{\mathsf{LAM}} M : A$ into a derivation $\mathcal{D}^\bullet \lhd \Gamma^\bullet \vdash_{\mathsf{IMLL}_2} M^\bullet : A^\bullet$:

- for all types $\sigma \in \Theta_\wedge$, it is defined as follows:

$$\begin{aligned} \alpha^\bullet &\triangleq \alpha \\ (A \multimap B)^\bullet &\triangleq A^\bullet \multimap B^\bullet \\ (A \wedge B)^\bullet &\triangleq A^\bullet \otimes B^\bullet \\ (\forall \alpha.A)^\bullet &\triangleq \forall \alpha.A^\bullet \end{aligned}$$

- for all contexts $\Gamma = x_1 : A_1, \dots, x_n : A_n$, we set $\Gamma^\bullet \triangleq x_1 : A_1^\bullet, \dots, x_n : A_n^\bullet$;
- for all typable terms $M \in \Lambda_{l,\wedge}$, it is defined as follows:

$$\begin{aligned} x^\bullet &\triangleq x \\ (\lambda x.N)^\bullet &\triangleq \lambda x.N^\bullet \end{aligned}$$

$$
\begin{aligned}
(NP)^\bullet &\triangleq N^\bullet P^\bullet \\
(N_1, N_2)^\bullet &\triangleq \langle N_1^\bullet, N_2^\bullet\rangle \\
(\texttt{proj}_1^{A_1\wedge A_2}(N))^\bullet &\triangleq \texttt{let } N^\bullet \texttt{ be } x_1, x_2 \texttt{ in } (\texttt{let } \texttt{E}_{A_2^\bullet}\, x_2 \texttt{ be } \mathbf{I} \texttt{ in } x_1) \\
(\texttt{proj}_2^{A_1\wedge A_2}(N))^\bullet &\triangleq \texttt{let } N^\bullet \texttt{ be } x_1, x_2 \texttt{ in } (\texttt{let } \texttt{E}_{A_1^\bullet}\, x_1 \texttt{ be } \mathbf{I} \texttt{ in } x_2) \\
(\texttt{copy}_A^W\, N \texttt{ as } x_1, x_2 \texttt{ in } (P_1, P_2))^\bullet &\triangleq \texttt{let } \texttt{D}_{A^\bullet}^{W^\bullet}\, N^\bullet \texttt{ be } x_1, x_2 \texttt{ in } \langle P_1^\bullet, P_2^\bullet\rangle.
\end{aligned}
$$

where Theorem 9 and Theorem 10 assure the existence of the erasure $\texttt{E}_{A_i^\bullet}$ and the duplicator $\texttt{D}_{A^\bullet}^{W^\bullet}$ of $\sigma^\bullet$ (with the notation as in Remark 4), because by Lemma 48.(1):

  - A, A_1, A_2 are closed and free from negative occurrences of $\forall$, so that $A^\bullet$, $A_1^\bullet$ and $A_2^\bullet$ are ground types,
  - W is an extended value that inhabits A, and hence $W^\bullet$ is a value that inhabits $A^\bullet$ by Lemma 48.(2);

- the definition of $(\_)^\bullet$ extends to any derivation $\mathcal{D} \lhd \Gamma \vdash M : A$ of LAM in the obvious way, following the structure of $M^\bullet$. Figure 4.5 collects the most interesting cases.

Before stating the simulation theorem we introduce a substitution lemma:

Lemma 49. *For all terms $M, N \in \Lambda_{l,\wedge}$ typable in* LAM*, $M^\bullet[N^\bullet/x] = (M[N/x])^\bullet$.*

Proof. Similar to Lemma 34. □

We now show that every reduction on terms typable in LAM can be simulated in the linear λ-calculus by means of the $\beta\eta$-reduction. Since by Theorem 3 every linear λ-term has type in IMLL, and hence in IMLL$_2$, this result can be seen as a simulation property relating LAM and IMLL$_2$.

Theorem 50 (Simulation for LAM)**.** *Let $\mathcal{D} \lhd \Gamma \vdash M : A$ be a derivation in* LAM*. If $M_1 \rightarrow^* M_2$ then $M_1^\bullet \rightarrow^*_{\beta\eta} M_2^\bullet$:*

$$\begin{tikzcd}
M_1 \arrow[r, "*"] \arrow[d, dashed] & M_2 \arrow[d, dashed] \\
M_1^\bullet \arrow[r, "*", "\beta\eta"'] & M_2^\bullet
\end{tikzcd}$$

Proof. By Theorem 46, it suffices to show that $M_1 \rightarrow M_2$ implies $M_1^\bullet \rightarrow^*_{\beta\eta} M_2^\bullet$. We proceed by case analysis and we consider the most interesting cases. Suppose M_1 is $(\lambda x.P)Q$ and $M_2 = P[Q/x]$. Lemma 49 implies $((\lambda y.P)Q)^\bullet = (\lambda y.P^\bullet)Q^\bullet \rightarrow_\beta P^\bullet[Q^\bullet/x] = (P[Q/x])^\bullet$. If M_1 is $\texttt{proj}_i^{A_1\wedge A_2}((W_1, W_2))$ and M_2 is W_i, then (W_1, W_2) is an extended value of type $A_1 \wedge A_2$ by Lemma 48.(1), hence $\langle W_1^\bullet, W_2^\bullet\rangle$ is a value of type $A_1^\bullet \otimes A_2^\bullet$ by Lemma 48.(2). Moreover, $\texttt{E}_{A_{3-i}^\bullet}$ is an eraser of $A_{3-i}^\bullet$ by Definition 43. Therefore:

$$
\begin{aligned}
(\texttt{proj}_i^{A_1\wedge A_2}((W_1, W_2)))^\bullet &= \texttt{let } \langle W_1^\bullet, W_2^\bullet\rangle \texttt{ be } x_1, x_2 \texttt{ in } (\texttt{let } \texttt{E}_{A_{3-i}^\bullet}\, x_{3-i} \texttt{ be } \mathbf{I} \texttt{ in } x_i) \\
&\rightarrow_\beta \texttt{let } \texttt{E}_{A_{3-i}^\bullet}\, W_{3-i} \texttt{ be } \mathbf{I} \texttt{ in } W_i \\
&\rightarrow^*_\beta W_i^\bullet
\end{aligned}
$$

by Theorem 9. If M_1 is $\texttt{copy}_A^{W'}\, W \texttt{ as } x_1, x_2 \texttt{ in } (N_1, N_2)$ and M_2 is $(N_1[W/x_1], N_2[W/x_2])$, then W is an extended value of type A by Lemma 48.(1), hence $W^\bullet$ is a value of type $A^\bullet$ by Lemma 48.(2). Moreover, $\texttt{D}_{A^\bullet}^{(W')^\bullet}$ is a duplicator of $A^\bullet$ by Definition 43. Therefore:

$$(\texttt{copy}_\sigma^{W'}\, W \texttt{ as } x_1, x_2 \texttt{ in } (N_1, N_2))^\bullet \triangleq \texttt{let } \texttt{D}_{\sigma^\bullet}^{(W')^\bullet}\, W^\bullet \texttt{ be } x_1, x_2 \texttt{ in } \langle N_1^\bullet, N_2^\bullet\rangle$$

$$\left(\dfrac{\begin{matrix}\mathcal{D}_1 \\ \vdash N_1 : A_1\end{matrix} \qquad \begin{matrix}\mathcal{D}_2 \\ \vdash N_2 : A_2\end{matrix}}{\vdash (N_1, N_2) : A_1 \wedge A_2}\,\wedge\mathrm{I0}\right)^{\bullet} \triangleq \dfrac{\left(\begin{matrix}\mathcal{D}_1 \\ \vdash N_1 : A_1\end{matrix}\right)^{\bullet} \qquad \left(\begin{matrix}\mathcal{D}_2 \\ \vdash N_2 : A_2\end{matrix}\right)^{\bullet}}{\vdash \langle N_1^{\bullet}, N_2^{\bullet}\rangle : A_1^{\bullet} \otimes A_2^{\bullet}}\,\otimes\mathrm{I}$$

$$\left(\dfrac{\begin{matrix}\mathcal{D} \\ \Gamma \vdash N : A_1 \wedge A_2\end{matrix} \qquad i \in \{1,2\}}{\Gamma \vdash \mathtt{proj}_i^{A_1 \wedge A_2}(N) : A_i}\,\wedge\mathrm{E}i\right)^{\bullet} \triangleq \dfrac{\left(\begin{matrix}\mathcal{D} \\ \Gamma \vdash N : A_1 \wedge A_2\end{matrix}\right)^{\bullet} \qquad \dfrac{\begin{matrix}\vdots \\ x_{3-i} : A_{3-i}^{\bullet} \vdash \mathtt{E}_{A_{3-i}^{\bullet}}\, x_{3-i} : \mathbf{1}\end{matrix} \qquad \dfrac{}{x_i : A_i^{\bullet} \vdash x_i : A_i^{\bullet}}\,ax}{x_1 : A_1^{\bullet}, x_2 : A_2^{\bullet} \vdash \mathtt{let}\ \mathtt{E}_{A_{3-i}^{\bullet}}\, x_{3-i}\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x_i : A_i^{\bullet}}\,\mathbf{1}\mathrm{E}}{\Gamma^{\bullet} \vdash \mathtt{let}\ N^{\bullet}\ \mathtt{be}\ x_1, x_2\ \mathtt{in}\ (\mathtt{let}\ \mathtt{E}_{A_{3-i}^{\bullet}}\, x_{3-i}\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ x_i) : A_i^{\bullet}}\,\otimes\mathrm{E}$$

$$\left(\dfrac{\begin{matrix}\mathcal{D}_1 \\ \Gamma \vdash N : A\end{matrix} \quad \begin{matrix}\mathcal{D}_2 \\ x_1 : A \vdash P_1 : B_1\end{matrix} \quad \begin{matrix}\mathcal{D}_3 \\ x_2 : A \vdash P_2 : B_2\end{matrix} \quad \begin{matrix}\mathcal{D}_4 \\ \vdash W : A\end{matrix}}{\Gamma \vdash \mathtt{copy}_A^W\ N\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (P_1, P_2) : B_1 \wedge B_1}\,\wedge\mathrm{I1}\right)^{\bullet} \triangleq$$

$$\dfrac{\dfrac{\begin{matrix}\left(\begin{matrix}\mathcal{D}_4 \\ \vdash W : A\end{matrix}\right)^{\bullet} \\ \vdots \\ \vdash \mathtt{D}_{A^{\bullet}}^{W^{\bullet}} : A^{\bullet} \multimap A^{\bullet} \otimes A^{\bullet}\end{matrix} \qquad \left(\begin{matrix}\mathcal{D}_1 \\ \Gamma \vdash N : A\end{matrix}\right)^{\bullet}}{\Gamma^{\bullet} \vdash \mathtt{D}_{A^{\bullet}}^{W^{\bullet}}\ N^{\bullet} : A^{\bullet} \otimes A^{\bullet}}\,\multimap\mathrm{E} \qquad \dfrac{\left(\begin{matrix}\mathcal{D}_2 \\ x_1 : A \vdash P_1 : B_1\end{matrix}\right)^{\bullet} \quad \left(\begin{matrix}\mathcal{D}_2 \\ x_2 : A \vdash P_2 : B_2\end{matrix}\right)^{\bullet}}{x_1 : A^{\bullet}, x_2 : A^{\bullet} \vdash \langle P_1^{\bullet}, P_2^{\bullet}\rangle : B_1^{\bullet} \otimes B_2^{\bullet}}\,\otimes\mathrm{I}}{\Gamma^{\bullet} \vdash \mathtt{let}\ \mathtt{D}_{A^{\bullet}}^{W^{\bullet}}\ N^{\bullet}\ \mathtt{be}\ x_1, x_2\ \mathtt{in}\ \langle P_1^{\bullet}, P_2^{\bullet}\rangle : B_1^{\bullet} \otimes B_2^{\bullet}}\,\otimes\mathrm{E}$$

Figure 4.5: The translation of the rules $\wedge$I0, $\wedge$Ei, and $\wedge$I1.

$$
\begin{aligned}
&\to^*_{\beta\eta} \texttt{let } \langle W^\bullet, W^\bullet \rangle \texttt{ be } x_1, x_2 \texttt{ in } \langle N_1^\bullet, N_2^\bullet \rangle && \text{Thm. 10}\\
&\to_\beta \langle N_1^\bullet[W^\bullet/x_1], N_2^\bullet[W^\bullet/x_2] \rangle\\
&= (N_1[W/x_1], N_2[W/x_2])^\bullet. && \text{Lem. 49}
\end{aligned}
$$

□

Theorem 51 (Exponential compression for LAM)**.** *Let $\mathcal{D} \lhd \Gamma \vdash M : A$ be a derivation in* LAM*. Then, $|M^\bullet| = \mathcal{O}(2^{|M|^k})$, for some $k \geq 1$.*

Proof. The proof is similar to the one of Theorem 37. □

4.2 The system STA$_\oplus$

In the previous section we presented LAM as an extension of IMLL$_2$ with *linear additives*, a weaker version of the usual additives that have no impact on the complexity of normalization. In what follows we study an application of linear additives to ICC by introducing STA$_\oplus$, a type system able to capture the probabilistic polynomial time (Theorems 76 and 91).

First, we recall Soft Linear Logic (SLL) [56], formulated as a type assignment for the standard λ-calculus, and we discuss a counterexample to the Subject reduction property for SLL. This leads us to consider a subsystem of SLL developed by Gaboardi and Ronchi Della Rocca in [40] and called *Soft Type Assignment* (STA).

Then, we present STA$_\oplus$ as a system combining STA with a non-deterministic variant of linear additives inspired by Diaz-Caro [33]. STA$_\oplus$ is defined as the type assignment for an extension of Simpson's linear λ-calculus [85] (see Section 2.2.3), whose terms will be endowed with a probabilistic multi-step reduction relation based on the surface reduction.

4.2.1 Soft Type Assignment

Soft Linear Logic (SLL) is a logical system introduced by Lafont in [56]. It is a subsystem of Second-Order Linear Logic (LL$_2$) in which the exponential rules p (*promotion*), d (*dereliction*), w (*weakening*), and c (*contraction*), displayed in intuitionistic form in Figure 4.6(a), are replaced by the rules sp (*soft promotion*) and m (*multiplexor*) in Figure 4.6(b). Since replacing the exponential rules with the rules sp, m and dig (*digging*) yields and equivalent formulation of Linear Logic, a fundamental aspect of SLL is that dig is forbidden and, as a consequence, that $!A \multimap !!A$ is no longer provable. This means that modalities in SLL can be used to keep track of the number of duplications in a derivation, producing a polynomial bound on the number of cut-elimination steps.

Following Gaboardi and Ronchi della Rocca [40], we present the intuitionistic version of SLL as a type assignment system for the standard λ-calculus in "quasi" natural deduction, i.e. all rules but those for "!" are in natural deduction style. We actually consider the $(\multimap, \forall, !)$-fragment of the system, being enough to capture PTIME. With a little abuse of terminology, we still refer to the resulting system as SLL.

Definition 44 (The system SLL)**.**

- Let $\mathcal{X}$ be a denumerable set of type variables. The *types* of SLL are generated by the following grammar:

$$A := \ \alpha \mid A \multimap A \mid !A \mid \forall\alpha.A \tag{4.9}$$

$$\frac{!A_1,\dots,!A_n \vdash C}{!A_1,\dots,!A_n \vdash !C}\,p \qquad \frac{\Gamma, A \vdash C}{\Gamma, !A \vdash C}\,d \qquad \frac{\Gamma \vdash C}{\Gamma, !A \vdash C}\,w \qquad \frac{\Gamma, !A, !A \vdash C}{\Gamma, !A \vdash C}\,c$$

(a) The exponential rules of Intuitionistic Linear Logic.

$$\frac{A_1,\dots,A_n \vdash C}{!A_1,\dots,!A_n \vdash !C}\,sp \qquad \frac{\Gamma, A, \overset{n}{\dots}, A \vdash C \qquad (n \geq 0)}{\Gamma, !A \vdash C}\,m \qquad \frac{\Gamma, !!A \vdash C}{\Gamma, !A \vdash C}\,dig$$

(b) An equivalent formulation of the exponential rules.

Figure 4.6: Linear Logic and exponential rules.

where $\alpha \in \mathcal{X}$. If Γ is $x_1 : A_1, \dots, x_n : A_n$ then $!\Gamma$ will be used on place of $x_1 : !A_1, \dots, x_n : !A_n$. With $A\langle B/\alpha\rangle$ we denote the standard meta-level substitution of B for every free occurrence of α in A.

- The system SLL, as a type assignment for the standard λ-calculus, is depicted in Figure 4.7(a).

It is well-known from Lincoln [62] that the subject reduction property fails in the type-assignment system obtained by decorating Intuitionistic Linear Logic with terms of the standard λ-calculus. As remarked in Gaboardi and Ronchi Della Rocca [40], this problem applies to SLL as well.

Example 16. A possible typing for the term $(\lambda y.yxx)((\lambda z.sz)w)$ in SLL is given by the derivation in Figure 4.7(b). After a step of β-reduction, we obtain the term $y((\lambda z.sz)w)((\lambda z.sz)w)$, which is no longer typable in the system.

Gaboardi and Ronchi della Rocca traced the failure of the subject reduction to the presence of modalities in the right-hand side of an implication, like in $A \multimap !B$, that are not introduced by a *sp* rule. To overcome this problem, they designed Soft Type Assignment (STA), a subsystem of SLL obtained by restricting types to the so-called *essential* ones, forbidding occurrences of the modality in the undesired positions.

Definition 45 (The system STA)**.**

- Let $\mathcal{X}$ be a denumerable set of type variables. The *essential types* are generated by the following grammar:

$$A := \ \alpha \mid \sigma \multimap A \mid \forall \alpha.A \tag{4.10}$$
$$\sigma := A \mid !\sigma \tag{4.11}$$

where $\alpha \in \mathcal{X}$. The grammar in (4.10) generates the *linear types*, and the grammar in (4.11) generates the *exponential types*. Exponential types are ranged over by σ, τ. If Γ is $x_1 : A_1, \dots, x_n : A_n$ then $!\Gamma$ will be used in place of $x_1 : !A_1, \dots, x_n : !A_n$. With $A\langle B/\alpha\rangle$ we denote the standard meta-level substitution of B for every free occurrence of α in A.

$$\dfrac{}{x : A \vdash x : A}\,ax$$

$$\dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B}\multimap\mathrm{I} \qquad \dfrac{\Gamma \vdash M : A \multimap B \qquad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B}\multimap\mathrm{E}$$

$$\dfrac{\Gamma \vdash M : A}{!\Gamma \vdash M : !A}\,sp \qquad \dfrac{\Gamma, x_1 : A, \overset{n}{\ldots}, x_n : A \vdash M : B \qquad (n \geq 0)}{\Gamma, x : !A \vdash M[x/x_1, \ldots, x/x_n] : B}\,m$$

$$\dfrac{\Gamma \vdash M : A\langle\gamma/\alpha\rangle \qquad \gamma \notin \mathrm{FV}(\Gamma)}{\Gamma \vdash M : \forall\alpha.A}\forall\mathrm{I} \qquad \dfrac{\Gamma \vdash M : \forall\alpha.A}{\Gamma \vdash M : A\langle B/\alpha\rangle}\forall\mathrm{E}$$

(a) The system SLL.

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{y : A \multimap A \multimap B, x_1 : A, x_2 : A \vdash yx_1x_2 : B}}{y : A \multimap A \multimap B, x : !A \vdash yxx : B}\,m}{y : A \multimap A \multimap B \vdash \lambda x.yxx : !A \multimap B}\multimap\mathrm{I} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{s : B \multimap !A \vdash s : B \multimap !A}\,ax \qquad \dfrac{}{z : B \vdash z : B}\,ax}{s : B \multimap !A, z : B \vdash sz : !A}\multimap\mathrm{E}}{s : B \multimap !A \vdash \lambda z.sz : B \multimap !A}\multimap\mathrm{I} \qquad \dfrac{}{w : B \vdash w : B}\,ax}{s : B \multimap !A, w : B \vdash (\lambda z.sz)w : !A}\multimap\mathrm{E}}{y : A \multimap A \multimap B, s : B \multimap !A, w : B \vdash (\lambda x.yxx)((\lambda z.sz)w) : B}\multimap\mathrm{E}$$

(b) Counterexample to the Subject reduction property in SLL.

Figure 4.7: The system SLL and the failure of the Subject reduction property.

$$\frac{}{x : A \vdash x : A}\ ax$$

$$\frac{\Gamma, x : \sigma \vdash M : A}{\Gamma \vdash \lambda x.M : \sigma \multimap A} \multimap\mathrm{I} \qquad \frac{\Gamma \vdash M : \sigma \multimap A \qquad \Delta \vdash N : \sigma}{\Gamma, \Delta \vdash MN : A} \multimap\mathrm{E}$$

$$\frac{\Gamma \vdash M : \sigma}{!\Gamma \vdash M : !\sigma}\ sp \qquad \frac{\Gamma, x_1 : \sigma, \overset{n}{\ldots}, x_n : \sigma \vdash M : \tau \qquad (n \geq 0)}{\Gamma, x : !\sigma \vdash M[x/x_1, \ldots, x/x_n] : \tau}\ m$$

$$\frac{\Gamma \vdash M : A\langle \gamma/\alpha \rangle \qquad \gamma \notin \mathrm{FV}(\Gamma)}{\Gamma \vdash M : \forall \alpha.A} \forall\mathrm{I} \qquad \frac{\Gamma \vdash M : \forall \alpha.A}{\Gamma \vdash M : A\langle B/\alpha \rangle} \forall\mathrm{E}$$

Figure 4.8: The system STA.

- STA is the type assignment system for the standard λ-calculus depicted in Figure 4.8.

A fundamental property of the system is that, whenever $\Gamma \vdash M : !\sigma$ is derivable in STA, the type $!\sigma$ must be introduced by a *sp* rule. This property is the key step to the subject reduction:

Proposition 52 (Subject reduction for STA [38]). *If $\mathcal{D} \lhd \Gamma \vdash M : \tau$ and $M \to_\beta M'$ then there exists $\mathcal{D}'$ such that $\mathcal{D}' \lhd \Gamma \vdash M' : \tau$.*

Theorem 53 (Polytime soundness and completeness [40]). *Every term M typable in STA can be evaluated to a normal form on a Turing Machine in polynomial time (with respect to $|M|$). Moreover, any language recognisable in polynomial time can be represented by a term with type in STA.*

A similar result has been proved in [40] for FPTIME, i.e. the class of functions on strings that can be computed in polynomial time (see Section 2.3.1).

4.2.2 The system STA$_\oplus$

In what follows we present STA$_\oplus$, a type system that extends STA with a non-deterministic variant of the linear additives in LAM, and we endow typable terms in the new system with a probabilistic multi-step reduction relation $\Rightarrow$ based on Simpson's surface reduction [85] (see Section 2.2.3).

To begin with, we define the types of STA$_\oplus$. These combine the essential types of STA with the linear additive connective $\wedge$.

Definition 46 (Types for STA$_\oplus$). Let $\mathcal{X}$ be a denumerable set of type variables. The grammar in (4.12) generates the *linear types* and the grammar in (4.13) the *exponential types*:

$$A, B := \alpha \mid \sigma \multimap A \mid A \wedge B \mid \forall \alpha.A \tag{4.12}$$

$$\sigma := A \mid !\sigma \tag{4.13}$$

where $\alpha \in \mathcal{X}$ and, in the clause $A \wedge B$ of (4.12), *both A and B must be closed and free from modalities and negative occurrences of* $\forall$. The set of all types generated by the grammar (4.13) is denoted $\Theta_{\wedge,!}$. A type is *strictly exponential* if it is of the form $!\sigma$. A *strictly exponential context* is a context containing only strictly exponential types and, similarly, a *linear context* contains only linear types. If Γ is $x_1 : A_1, \dots, x_n : A_n$, then $!\Gamma$ is $x_1 : !A_1, \dots, x_n : !A_n$. Finally, $A\langle B/\alpha\rangle$ is the standard meta-level substitution of B for every occurrence of α in A.

We shall define $\mathsf{STA}_{\oplus}$ as the type assignment system for the term calculus $\Lambda^{!}_{l,\oplus}$ that can be described as follows:

- it is based on Simpson's *linear λ-calculus* [85], i.e. it has a linear abstraction $\lambda x.M$, a non-linear abstraction $\lambda !x.M$, a !-operator $!N$, and it is endowed with a surface reduction not evaluating inside the scope of "!" (see Section 2.2.3);
- it has explicit dereliction $\mathtt{d}$, as in Ronchi Della Rocca and Roversi [77];
- it has the construct $\mathtt{copy}^{W}_{A}$ and pairs (M, N) from LAM;
- it has constructs of the form $\mathsf{proj}^{A\wedge B}_{A}$ and $\mathsf{proj}^{A\wedge B}_{B}$, which are type-dependent variants of the projections $\mathsf{proj}^{A\wedge B}_{1}$ and $\mathsf{proj}^{A\wedge B}_{2}$ of LAM based on Diaz-Caro [33].

The !-operator and the related surface reduction will play a central role in recovering confluence in a probabilistic setting (Section 4.3.1). The presence of explicit dereliction constructs $\mathtt{d}$ is then required to assure Subject reduction (Theorem 72), as shown in Remark 11.

Intuitively, the construct $\mathsf{proj}^{A\wedge B}_{C}$ of the latter point selects the component of a pair with type $C \in \{A, B\}$. This type-dependency let the non-determinism arise naturally: whenever a term of the form $\mathsf{proj}^{A\wedge A}_{A}((W_1, W_2))$ is reached during the surface reduction, the lack of information about which component of the pair to select forces a non-deterministic choice in order to break the "stalemate", as anticipated in Remark 8.

Definition 47 (The calculi $\Lambda^{!}_{\oplus}$ and $\Lambda^{!}_{l,\oplus}$)**.**

- An *extended value* is any term in Λ_{π} that is either a closed and normal term of Λ_l or a pair (M, N), with M and N extended values. Extended values are ranged over by W.
- Let $\mathcal{V}$ be a denumerable set of variables. The set $\Lambda^{!}_{\oplus}$ of *terms* is generated by the following grammar:
$$\begin{aligned} M := x \mid \lambda x.M \mid \lambda !x.M \mid MM \mid\, !M \mid \mathtt{d}(M) \mid (M, M) \\ \mathtt{proj}^{A\wedge B}_{C}(M) \mid \mathtt{copy}^{W}_{A}\, M \;\mathtt{as}\; x, y \;\mathtt{in}\; (M, M). \end{aligned} \tag{4.14}$$
where $x, y \in \mathcal{V}$, $C \in \{A, B\}$, W is an extended value, and $A, B \in \Theta_{\wedge,!}$. With $!^n M$ (resp. $\mathtt{d}^n(M)$) we denote $!\overset{n}{\dots}!M$ (resp. $\mathtt{d}(\overset{n}{\dots}\mathtt{d}(M)\dots)$). We extend both $FV(M)$ and $|M|$ in Definition 41 to the new clauses as follows:
$$\begin{aligned} \mathrm{FV}(\lambda !x.M) &= \mathrm{FV}(M) \setminus \{x\} & |\lambda !x.M| &= |M| + 1 \\ \mathrm{FV}(!M) = \mathrm{FV}(\mathtt{d}(M)) &= \mathrm{FV}(M) & |!M| = |\mathtt{d}(M)| &= |M| + 1. \end{aligned}$$
- The meta-level substitution of N for the free occurrences of x in M, written $M[N/x]$, is defined as usual. We also define a new substitution, called *forgetful substitution* and written $M\{N/x\}$, as follows:
$$M\{N/x\} \triangleq \begin{cases} M'\{N'/x'\} & \text{if } N = !N' \text{ and } M = M'[\mathtt{d}(x)/x'], \text{ with } x \notin \mathrm{FV}(M'), \\ M[N/x] & \text{otherwise.} \end{cases}$$
As usual, $M\{N/x_1, \dots, N/x_n\}$ denotes $((M\{N/x_1\})\dots)\{N/x_n\}$.

- A *surface context* is a term in $\Lambda^!_\oplus$ containing a unique hole $[\cdot]$, generated by the following grammar:

$$\begin{aligned}\mathcal{C} := [\cdot] \mid \lambda x.\mathcal{C} \mid \lambda !x.\mathcal{C} \mid \mathcal{C}M \mid M\mathcal{C} \mid \mathtt{d}(\mathcal{C}) \mid (\mathcal{C}, M) \mid (M, \mathcal{C}) \mid \mathtt{proj}^{A_1\wedge A_2}_{A_i}(\mathcal{C})\\ \mathtt{copy}^V_A\, \mathcal{C} \;\mathtt{as}\; x_1, x_2 \;\mathtt{in}\; (M_1, M_2) \mid \mathtt{copy}^V_A\, M \;\mathtt{as}\; x_1, x_2 \;\mathtt{in}\; (\mathcal{C}, N)\\ \mathtt{copy}^V_A\, M \;\mathtt{as}\; x_1, x_2 \;\mathtt{in}\; (N, \mathcal{C}).\end{aligned}$$

  If $\mathcal{C}$ is a surface context and M is a term, then $\mathcal{C}[M]$ denotes the term obtained by substituting the unique hole in $\mathcal{C}$ with possible capture of free variables in M.

- The *one-step surface reduction relation* $\to$ between terms in $\Lambda^!_\oplus$ and pair of terms in $\Lambda^!_\oplus$ is defined by the following rules that apply to any surface context:

$$\begin{aligned}(\lambda x.M)N &\to M[N/x] \\ (\lambda !x.M)!N &\to M\{!N/x\} \\ \mathtt{proj}^{A\wedge B}_A(W_1, W_2) &\to W_1 && A \neq B \\ \mathtt{proj}^{A\wedge B}_B(W_1, W_2) &\to W_2 && A \neq B \\ \mathtt{proj}^{A\wedge A}_A(W_1, W_2) &\to W_1, W_2 \\ \mathtt{copy}^{W'}_A\, W \;\mathtt{as}\; x_1, x_2 \;\mathtt{in}\; (M_1, M_2) &\to (M_1[W/x_1], M_2[W/x_2]).\end{aligned} \tag{4.15}$$

  where $M \to N$ is shorthand for $M \to N, N$. A term is in (or is a) *surface normal form* if no reduction applies to it. Surface normal forms are ranged over by S.

- Let $M \in \Lambda^!_\oplus$. A variable is *surface-linear* (also *s-linear*) *in* M if x occurs exactly once in M and, moreover, this free occurrence of x lies neither within the scope of a !-operator nor within the scope of a $\mathtt{d}$-operator in M. We say that M is *surface-linear* (also *s-linear*) if:

  - for each subterm $\lambda x.N$ of M, x is s-linear in N;
  - for each subterm (N_1, N_2) of M not in the scope of a $\mathtt{copy}^W_A$ construct, $FV(N_1) = FV(N_2) = \emptyset$;
  - for each subterm $\mathtt{copy}^{W'}_A\, N \;\mathtt{as}\; x_1, x_2 \;\mathtt{in}\; (P_1, P_2)$ of M, x_i is s-linear in P_i and $FV(P_1) = \{x_1\} \neq \{x_2\} = FV(P_2)$.

  The set of all s-linear terms in $\Lambda^!_\oplus$ is denoted $\Lambda^!_{l,\oplus}$. The set of all surface normal forms in $\Lambda^!_{l,\oplus}$ will be denoted SNF.

Note that the extended values are terms in $\Lambda^!_{l,\oplus}$. The following proposition says that reducing s-linear terms yields s-linear terms:

Proposition 54. *If $M \in \Lambda^!_{l,\oplus}$ and $M \to M'$ then $M' \in \Lambda^!_{l,\oplus}$.*

Remark 10. Because of the presence of an explicit dereliction $\mathtt{d}$, to define the surface reduction step for the redex $(\lambda !x.M)!N$ in (4.15) we introduced a further meta-level substitution, $M\{N/x\}$, that we called “forgetful” in Definition 47. As compared to the corresponding surface reduction step $(\lambda !x.M)!N \to_! M[N/x]$ in Simpson’s linear λ-calculus [85] (see Section 2.2.3), the rule $(\lambda !x.M)!N \to M\{!N/x\}$ involves a more general mechanism that consumes several occurrences of both the operators ! and $\mathtt{d}$ at a time. As an example, the following:

$$(\lambda !x.z\, \mathtt{d}^3(x)\, \mathtt{d}(x))(!^2 y) \to (z\, \mathtt{d}^3(x)\, \mathtt{d}^2(x))\{(!^2 y)/x\} \quad \left(z\, \mathtt{d}^3(x)\, \mathtt{d}^2(x) \overset{def}{=} (z\, \mathtt{d}^2(x')\, \mathtt{d}(x'))[\mathtt{d}(x)/x']\right)$$

$$\frac{S \in \mathrm{SNF}}{S \Rightarrow S}\, s1 \qquad \frac{M \to M_1, M_2 \qquad M_1 \Rightarrow \mathscr{D}_1 \qquad M_2 \Rightarrow \mathscr{D}_2}{M \Rightarrow \dfrac{1}{2} \cdot \mathscr{D}_1 + \dfrac{1}{2} \cdot \mathscr{D}_2}\, s2$$

Figure 4.9: Multi-step reduction $\Rightarrow$ for $\Lambda^!_\oplus$.

$$\begin{aligned}
&= (z\,\mathtt{d}^2(x')\,\mathtt{d}(x'))\{(!y)/x'\} \qquad \left(z\,\mathtt{d}^2(x')\,\mathtt{d}(x') \overset{def}{=} (z\,\mathtt{d}(x'')\,x'')[\mathtt{d}(x')/x'']\right)\\
&= (z\,\mathtt{d}(x'')\,x'')\{y/x''\}\\
&= z\,\mathtt{d}(y)\,y.
\end{aligned}$$

shows an application of this rule to the term $(\lambda !x.z\,\mathtt{d}^3(x)\,\mathtt{d}(x))(!^2y)$.

Informally, $M \to M_1, M_2$ means that M can be evaluated in one step to both M_1 and M_2 with equal probability $\frac{1}{2}$. We now endow $\Lambda^!_\oplus$ with a probabilistic multi-step reduction relation giving a formal counterpart to this intuition. This relation cannot be defined by simply taking the reflexive and transitive closure of $\to$, since a term can reduce in a given number of steps to several terms with distinct probabilities. The solution is to define a relation between a term M and a distribution $\mathscr{D}$ of surface normal forms, the latter representing the set of all possible outcomes of the evaluation of M, weighted with the probability of obtaining them.

Definition 48 (Multi-step reduction $\Rightarrow$)**.**

- A *surface distribution* is a probability distribution over SNF, i.e. a function $\mathscr{D} : \mathrm{SNF} \longrightarrow [0, 1]$ such that:
$$\sum_{S \in \mathrm{SNF}} \mathscr{D}(S) = 1.$$
- The *multi-step reduction* $\Rightarrow$ is the relation between terms of $\Lambda^!_\oplus$ and surface distributions, defined by the rules in Figure 4.9.
- We inductively define the *size* $|\pi|$ of a derivation $\pi : M \Rightarrow \mathscr{D}$ as follows:
  - if the last rule of π is $s1$ then $|\pi| \triangleq 0$;
  - otherwise, if the last rule of π is $s2$:
$$\frac{M \to M_1, M_2 \qquad \pi_1 : M_1 \Rightarrow \mathscr{D}_1 \qquad \pi_2 : M_2 \Rightarrow \mathscr{D}_2}{M \Rightarrow \frac{1}{2} \cdot \mathscr{D}_1 + \frac{1}{2} \cdot \mathscr{D}_2}\, s2$$
    then $|\pi| \triangleq \max(|\pi_1|, |\pi_2|) + 1$.

In Section 2.5.2 we introduced the basic definitions and conventions related to probability distributions and relations. So, for example, given $\{S_1, \ldots, S_n\} \subseteq \mathrm{SNF}$ and $r_1, \ldots, r_n \in [0, 1]$ such that $\sum_{i=1}^n r_i = 1$, the expression $r_1 \cdot S_1 + \ldots + r_n \cdot S_n$ means a distribution $\mathscr{D}$ such that $\mathscr{D}(S_i) = r_i$, for all $i \leq n$. Then, S can denote both a surface normal form and a surface distribution having all its mass in S. Moreover, given two surface distributions $\mathscr{D}_1$ and $\mathscr{D}_2$, with

$\frac{1}{2}\cdot\mathscr{D}_1+\frac{1}{2}\cdot\mathscr{D}_2$ we denote the surface distribution defined, for all $S \in \text{SNF}$, by $(\frac{1}{2}\cdot\mathscr{D}_1+\frac{1}{2}\cdot\mathscr{D}_2)(S) = \frac{1}{2}\cdot\mathscr{D}_1(S)+\frac{1}{2}\cdot\mathscr{D}_2(S)$. Finally, derivations of $M \Rightarrow \mathscr{D}$ are ranged over π, ρ.

In the following we show a derivation of a multi-step reduction. Other derivations can be found in Example 18.

Example 17. Consider the term $(\mathbf{T}\oplus\mathbf{F})\mathbf{TF}$, where $\mathbf{T} \triangleq \lambda xy.x$, $\mathbf{F} \triangleq \lambda xy.y$ and $\mathbf{T}\oplus\mathbf{F} \triangleq \mathtt{proj}_A^{A\wedge A}(\mathbf{T},\mathbf{F})$, for some type A. This term reduces to the surface normal forms $\mathbf{T}$ and $\mathbf{F}$. Indeed, $(\mathbf{T}\oplus\mathbf{F})\mathbf{TF} \to \mathbf{TTF},\mathbf{FTF}$, with $\mathbf{TTF} \to (\lambda y.\mathbf{T})\mathbf{F} \to \mathbf{T}$ and $\mathbf{FTF} \to (\lambda y.y)\mathbf{F} \to \mathbf{F}$, where we recall that $M \to N$ stands for $M \to N, N$. The related multi-step reduction $(\mathbf{T}\oplus\mathbf{F})\mathbf{TF} \Rightarrow \frac{1}{2}\cdot\mathbf{T}+\frac{1}{2}\cdot\mathbf{F}$ is the following:

$$\dfrac{(\mathbf{T}\oplus\mathbf{F})\mathbf{TF} \to \mathbf{TTF},\mathbf{FTF} \qquad \dfrac{\mathbf{TTF} \to (\lambda y.\mathbf{T})\mathbf{F} \qquad \dfrac{(\lambda y.\mathbf{T})\mathbf{F} \to \mathbf{T} \qquad \dfrac{}{\mathbf{T} \Rightarrow \mathbf{T}}\,s1}{(\lambda y.\mathbf{T})\mathbf{F} \Rightarrow \mathbf{T}}\,s2}{\mathbf{TTF} \Rightarrow \mathbf{T}}\,s2 \qquad \begin{array}{c}\vdots\\ \mathbf{FTF} \Rightarrow \mathbf{F}\end{array}}{(\mathbf{T}\oplus\mathbf{F})\mathbf{TF} \Rightarrow \frac{1}{2}\cdot\mathbf{T}+\frac{1}{2}\cdot\mathbf{F}}\,s2$$

We can now define the system $\mathsf{STA}_\oplus$ as follows:

Definition 49 (The system $\mathsf{STA}_\oplus$)**.** $\mathsf{STA}_\oplus$ is the type assignment system (in natural deduction style) for the term calculus $\Lambda^!_{l,\oplus}$ displayed in Figure 4.10, where the notation $\lambda(!)x.M$ stands for $\lambda!x.M$ if σ is strictly exponential, and $\lambda x.M$ otherwise. The system requires two conditions:

- the type A in $\wedge$I1 *must be closed and free from modalities and negative occurrences of* $\forall$;
- *Context condition*: modalities cannot occur in the contexts of the rules $\wedge$I1 and $\wedge$E.

Let us remark that both conditions we imposed in $\mathsf{STA}_\oplus$ imply that:

Fact 55. *Every type appearing in the rules $\wedge$I0, $\wedge$I1 and $\wedge$E is free from modalities.*

The above fact enables us to distinguish two "layers" in a term typable in $\mathsf{STA}_\oplus$: a low-level layer concerning the linear additives, where restricted forms of duplication and erasure for finite data types are permitted, and a top-level layer concerning the exponential rules, where duplication and erasure can be performed unconditionally.

Remark 11. The introduction of a !-operator requires the use of explicit dereliction in the term calculus to assure Subject reduction (Theorem 72). Indeed, by ruling out all instances of $\mathtt{d}$ in the inference rules of $\mathsf{STA}_\oplus$ one can construct the following derivation:

$$\dfrac{\dfrac{\dfrac{}{x : A \vdash x : A}\,ax}{\overline{x : !!A \vdash !!x : !!A}}\,sp \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y_1 : A \vdash y_1 : A}\,ax \qquad \dfrac{}{y_2 : A \vdash y_2 : A}\,ax}{y_1 : A, y_2 : A \vdash \langle y_1, y_2\rangle : A\otimes A}\,\otimes\text{R}}{y : !A \vdash \langle y, y\rangle : A\otimes A}\,m}{z : !!A \vdash \langle z, z\rangle : A\otimes A}\,m}{\vdash \lambda!z.\langle z, z\rangle : !!A \multimap A\otimes A}\,\multimap\text{I}}{x : !!A \vdash (\lambda!z.\langle z, z\rangle)!!x : A\otimes A}\,\multimap\text{E}$$

If we apply to $(\lambda!z.\langle z, z\rangle)!!x$ the surface reduction step $(\lambda!x.M)!N \to_! M[N/x]$ in Simpson's linear λ-calculus (see Section 2.2.3), we would obtain $x : !!A \vdash \langle !x, !x\rangle : A\otimes A$. However, one can easily check that this judgement cannot be derived in the system.

We conclude with the following property analogous to Proposition 43:

$$\frac{}{x : A \vdash x : A}\ ax$$

$$\frac{\Gamma, x : \sigma \vdash M : B}{\Gamma \vdash \lambda(!)x.M : \sigma \multimap B}\ \multimap\mathrm{I} \qquad \frac{\Gamma \vdash M : \sigma \multimap B \qquad \Delta \vdash N : \sigma}{\Gamma, \Delta \vdash MN : B}\ \multimap\mathrm{E}$$

$$\frac{\vdash M_1 : B_1 \qquad \vdash M_2 : B_2}{\vdash (M_1, M_2) : B_1 \wedge B_2}\ \wedge\mathrm{I0} \qquad \frac{\Gamma \vdash M : A \wedge B \qquad C \in \{A, B\}}{\Gamma \vdash \mathtt{proj}_C^{A \wedge B}(M) : C}\ \wedge\mathrm{E}$$

$$\frac{\Gamma \vdash N : A \qquad x_1 : A \vdash M_1 : B_1 \qquad x_2 : A \vdash M_2 : B_2 \qquad \vdash W : A}{\Gamma \vdash \mathtt{copy}_A^W\ N\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (M_1, M_2) : B_1 \wedge B_2}\ \wedge\mathrm{I1}$$

$$\frac{x_1 : \sigma_1, \dots, x_n : \sigma_n \vdash M : \tau}{y_1 : !\sigma_1, \dots, y_n : !\sigma_n \vdash !M[\mathtt{d}(y_1)/x_1, \dots, \mathtt{d}(y_n)/x_n] : !\tau}\ sp$$

$$\frac{\Gamma, x_1 : \sigma, \dots, x_n : \sigma \vdash M : \tau \qquad (n \geq 0)}{\Gamma, x : !\sigma \vdash M[\mathtt{d}(x)/x_1, \dots, \mathtt{d}(x)/x_n] : \tau}\ m$$

$$\frac{\Gamma \vdash M : A\langle \gamma/\alpha \rangle \qquad \gamma \notin \mathrm{FV}(\Gamma)}{\Gamma \vdash M : \forall \alpha.A}\ \forall\mathrm{I} \qquad \frac{\Gamma \vdash M : \forall \alpha.A}{\Gamma \vdash M : A\langle B/\alpha \rangle}\ \forall\mathrm{E}$$

Figure 4.10: The system $\mathsf{STA}_\oplus$.

Proposition 56. *Let W be an extended value and let A be a closed type free from modalities and negative occurrences of $\forall$. If $\mathcal{D} \triangleleft \vdash W : A$ in $\mathsf{STA}_\oplus$, then $|W| \leq |A|$.*

Proof. It suffices to prove by an easy induction on the structure of terms that, for all normal terms $M \in \Lambda^!_{l,\oplus}$ such that $x_1 : A_1, \dots, x_n : A_n \vdash M : A$, and A (resp. $A_1, \dots, A_n$) free from ! and from negative occurrences (resp. positive occurrences) of $\forall$, then $|M| \leq \sum_{i=1}^n |A_i| + |A|$. □

4.3 Polytime soundness

In this section we prove that $\mathsf{STA}_\oplus$ is sound with respect to the polytime Probabilistic Turing Machines. We essentially adapt to the probabilistic setting the proof techniques developed for STA in [40], which are well-known since [56].

First, following essentially the same approach of Dal Lago and Toldin in [28], we prove the confluence of $\Rightarrow$, i.e. the uniqueness of distributions, for terms in $\Lambda^!_{l,\oplus}$ (Theorem 64).

Then, we define a notion of "weight" for derivations, and prove a stronger formulation of the Subject reduction property (Theorem 72), stating that the surface reduction preserves the type and shrinks the weight of derivations.

This result allows us to polynomially bound the number of the surface reduction steps from a typable term to surface normal forms (Lemma 75), and hence to prove the Polytime Soundness

Theorem (Theorem 76).

4.3.1 Confluence

Probabilistic calculi are not confluent, because a term may reduce to several terms in a single step. A confluence property can be somehow recovered by looking at the distributions of possible observables, in our case the surface normal forms, and by proving that a unique distribution can be associated with each term. However, this is not enough in general, as the example below shows.

Example 18. Consider the following term in the calculus $\Lambda^!_\oplus$:

$$\mathtt{coin} \triangleq (\lambda x.\langle x, x\rangle)\,\mathtt{ran} \tag{4.16}$$

$$\mathtt{ran} \triangleq \mathtt{proj}^{\mathbf{B}\wedge\mathbf{B}}_{\mathbf{B}}(\mathtt{tt}, \mathtt{ff}) \tag{4.17}$$

where $\mathbf{B}$, $\mathtt{tt}$, and $\mathtt{ff}$ are as in (3.2), and $\langle M, N\rangle$ is as in Definition 3. Clearly, $\mathtt{coin} \notin \Lambda^!_{l,\oplus}$ because $\lambda x.\langle x, x\rangle$ is not s-linear. There are two different surface reductions for this term: we can first reduce $\mathtt{ran}$ and then apply $\lambda x.\langle x, x\rangle$ to the result in order to obtain either $\langle \mathtt{tt}, \mathtt{tt}\rangle$ or $\langle \mathtt{ff}, \mathtt{ff}\rangle$, or we can first apply $\lambda x.\langle x, x\rangle$ to $\mathtt{ran}$ and then reduce each separate copy of $\mathtt{ran}$, obtaining one among $\langle \mathtt{tt}, \mathtt{tt}\rangle$, $\langle \mathtt{tt}, \mathtt{ff}\rangle$, $\langle \mathtt{ff}, \mathtt{tt}\rangle$, and $\langle \mathtt{ff}, \mathtt{ff}\rangle$. Formally, we have:

$$\mathtt{coin} \Rightarrow \frac{1}{2}\cdot\langle \mathtt{tt}, \mathtt{tt}\rangle + \frac{1}{2}\cdot\langle \mathtt{ff}, \mathtt{ff}\rangle$$

$$\mathtt{coin} \Rightarrow \frac{1}{4}\cdot\langle \mathtt{tt}, \mathtt{tt}\rangle + \frac{1}{4}\cdot\langle \mathtt{tt}, \mathtt{ff}\rangle + \frac{1}{4}\cdot\langle \mathtt{ff}, \mathtt{tt}\rangle + \frac{1}{4}\cdot\langle \mathtt{ff}, \mathtt{ff}\rangle$$

whose derivations are in Figure 4.11(a).

The example above shows that there is no chance of associating a unique distribution to terms in $\Lambda^!_\oplus$. But what about its subset $\Lambda^!_{l,\oplus}$? We start by considering a s-linear variant of $\mathtt{coin}$ in the following example.

Example 19. The term $\mathtt{coin}_! \triangleq (\lambda !x.\langle x, x\rangle)\,!\mathtt{ran}$ is s-linear, hence $\mathtt{coin}_! \in \Lambda^!_{l,\oplus}$. Moreover, it is typable in $\mathsf{STA}_\oplus$, as shown in Figure 4.11(b). We observe that $\mathtt{ran}$ is in the scope of "!", where surface reduction is forbidden. This means that we are forced to apply $\lambda !x.\langle x, x\rangle$ to $\mathtt{ran}$ before evaluating the latter, thus obtaining $\frac{1}{4}\cdot\langle \mathtt{tt}, \mathtt{tt}\rangle + \frac{1}{4}\cdot\langle \mathtt{tt}, \mathtt{ff}\rangle + \frac{1}{4}\cdot\langle \mathtt{ff}, \mathtt{tt}\rangle + \frac{1}{4}\cdot\langle \mathtt{ff}, \mathtt{ff}\rangle$ as the unique distribution.

Example 19 suggests that s-linearity can be a sufficient condition on $\Lambda^!_\oplus$ to ensure the confluence of $\Rightarrow$, i.e. that for every term $M \in \Lambda^!_{l,\oplus}$ there is *at most one* distribution $\mathscr{D}$ such that $M \Rightarrow \mathscr{D}$. We will show that this is the case by adapting the techniques in Dal Lago and Toldin [28].

The first step toward confluence is to show that $\rightarrow$ enjoys a strong confluence property on $\Lambda^!_{l,\oplus}$:

Lemma 57. *Let $M, N \in \Lambda^!_{l,\oplus}$:*

(1) If $M \rightarrow M', M''$ then $M\{N/x\} \rightarrow M'\{N/x\}, M''\{N/x\}$

(2) If $N \rightarrow N', N''$ and x is linear in M then $M[N/x] \rightarrow M[N'/x], M[N''/x]$.

Proof. Easy induction on the structure of M. □

$$\dfrac{\texttt{coin} \to (\lambda x.\langle x,x\rangle)\,\texttt{tt}\,,\,(\lambda x.\langle x,x\rangle)\,\texttt{ff} \qquad \dfrac{(\lambda x.\langle x,x\rangle)\,\texttt{tt} \to \texttt{tt}^2 \quad \dfrac{}{\texttt{tt}^2 \Rightarrow \texttt{tt}^2}}{(\lambda x.\langle x,x\rangle)\,\texttt{tt} \Rightarrow \texttt{tt}^2} \quad \dfrac{(\lambda x.\langle x,x\rangle)\,\texttt{ff} \to \texttt{ff}^2 \quad \dfrac{}{\texttt{ff}^2 \Rightarrow \texttt{ff}^2}}{(\lambda x.\langle x,x\rangle)\,\texttt{ff} \Rightarrow \texttt{ff}^2}}{\texttt{coin} \Rightarrow \frac{1}{2}\cdot\texttt{tt}^2 + \frac{1}{2}\cdot\texttt{ff}^2}$$

$$\dfrac{\texttt{coin} \to \texttt{ran}^2 \qquad \dfrac{\texttt{ran}^2 \to \langle\texttt{tt},\texttt{ran}\rangle,\langle\texttt{ff},\texttt{ran}\rangle \quad \dfrac{\langle\texttt{tt},\texttt{ran}\rangle \to \texttt{tt}^2,\langle\texttt{tt},\texttt{ff}\rangle \quad \dfrac{}{\texttt{tt}^2 \Rightarrow \texttt{tt}^2} \quad \dfrac{}{\langle\texttt{tt},\texttt{ff}\rangle \Rightarrow \langle\texttt{tt},\texttt{ff}\rangle}}{\langle\texttt{tt},\texttt{ran}\rangle \Rightarrow \frac{1}{2}\cdot\texttt{tt}^2 + \frac{1}{2}\cdot\langle\texttt{tt},\texttt{ff}\rangle} \quad \begin{array}{c}\vdots\\ \langle\texttt{ff},\texttt{ran}\rangle \Rightarrow \frac{1}{2}\cdot\langle\texttt{ff},\texttt{tt}\rangle + \frac{1}{2}\cdot\texttt{ff}^2\end{array}}{\texttt{ran}^2 \Rightarrow \frac{1}{4}\cdot\texttt{tt}^2 + \frac{1}{4}\cdot\langle\texttt{tt},\texttt{ff}\rangle + \frac{1}{4}\cdot\langle\texttt{ff},\texttt{tt}\rangle + \frac{1}{4}\cdot\texttt{ff}^2}}{\texttt{coin} \Rightarrow \frac{1}{4}\cdot\texttt{tt}^2 + \frac{1}{4}\cdot\langle\texttt{tt},\texttt{ff}\rangle + \frac{1}{4}\cdot\langle\texttt{ff},\texttt{tt}\rangle + \frac{1}{4}\cdot\texttt{ff}^2}$$

(a) Counterexample to confluence for $\Lambda^!_\oplus$, where $M^2 \triangleq \langle M, M\rangle$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x_1 : \mathbf{B} \vdash x_1 : \mathbf{B}}\,ax \quad \dfrac{}{x_2 : \mathbf{B} \vdash x_2 : \mathbf{B}}\,ax}{x_1 : \mathbf{B}, x_2 : \mathbf{B} \vdash \langle x_1, x_2\rangle : \mathbf{B}\otimes\mathbf{B}}\,\otimes\mathrm{I}}{x : !\mathbf{B} \vdash \langle x, x\rangle : \mathbf{B}\otimes\mathbf{B}}\,m}{\vdash \lambda !x.\langle x,x\rangle : !\mathbf{B} \multimap \mathbf{B}\otimes\mathbf{B}}\,\multimap\mathrm{I} \qquad \dfrac{\dfrac{\dfrac{\vdash \texttt{tt} : \mathbf{B} \qquad \vdash \texttt{ff} : \mathbf{B}}{\vdash (\texttt{tt},\texttt{ff}) : \mathbf{B}\wedge\mathbf{B}}\,\wedge\mathrm{I0}}{\vdash \texttt{proj}^{\mathbf{B}\wedge\mathbf{B}}_{\mathbf{B}}((\texttt{tt},\texttt{ff})) : \mathbf{B}}\,\wedge\mathrm{E}}{\vdash !(\texttt{proj}^{\mathbf{B}\wedge\mathbf{B}}_{\mathbf{B}}((\texttt{tt},\texttt{ff}))) : !\mathbf{B}}\,sp}{\vdash (\lambda !x.\langle x,x\rangle)\,!(\texttt{proj}^{\mathbf{B}\wedge\mathbf{B}}_{\mathbf{B}}((\texttt{tt},\texttt{ff}))) : \mathbf{B}\otimes\mathbf{B}}\,\multimap\mathrm{E}$$

(b) A derivation of $\texttt{coin}_! \triangleq (\lambda !x.\langle x,x\rangle)\,!(\texttt{proj}^{\mathbf{B}\wedge\mathbf{B}}_{\mathbf{B}}((\texttt{tt},\texttt{ff})))$.

Figure 4.11: The terms coin and $\texttt{coin}_!$.

Lemma 58. *Let $M \in \Lambda^{!}_{l,\oplus}$. If $M \to M'$ and $M \to M''$, with M' and M'' distinct, then there exists a term N such that $M' \to N$ and $M'' \to N$.*

Proof. By induction on the structure of M. We just consider the most interesting cases. If $M = (\lambda x.P)Q \to P[Q/x] = M'$, then either $M'' = (\lambda x.P')Q$ with $P \to P'$ or $M'' = (\lambda x.P)Q'$ with $Q \to Q'$. Since M is s-linear, x is s-linear in P and hence x does not lie within the scope of a $\mathtt{d}$-operator. This means that $P[Q/x] = P\{Q/x\}$ by definition. In the first case, we have $M' \to P'[Q/x]$ by Lemma 57.(1) and also $M'' \to P'[Q/x]$. In the second case, we have $M' \to P[Q'/x]$ by Lemma 57.(2), and also $M'' \to P[Q'/x]$. Similarly, if $M = (\lambda !x.P)!Q \to P\{!Q/x\} = M'$ then the only case is $M'' = (\lambda !x.P')!Q$ where $P \to P'$, since reduction is forbidden in Q. By Lemma 57.(1), $M' \to P'\{!Q/x\}$, and also $M'' \to P'\{!Q/x\}$. Last, we consider the case where $M = \mathtt{copy}^{W'}_{A}\ W\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (N_1, N_2)$, $M' = (N_1[W/x_1], N_2[W/x_2])$, and $M'' = \mathtt{copy}^{W'}_{A}\ W\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (N_1', N_2)$. Since M is s-linear, x_1 is s-linear in N_1 and hence x does not lie within the scope of a $\mathtt{d}$-operator. This means that $N_1[W/x] = N_1\{W/x\}$ by definition. Then $M' \to (N_1'[W/x_1], N_2[W/x_2])$ by Lemma 57.(1) and also $M'' \to (N_1'[W/x_1], N_2[W/x_2])$. □

Lemma 59. *Let $M \in \Lambda^{!}_{l,\oplus}$. If $M \to M_1', M_2'$ and $M \to M''$, with M_1' and M_2' distinct, then there exist terms N_1 and N_2 such that $M_1' \to N_1$, $M_2' \to N_2$ and $M'' \to N_1, N_2$.*

Proof. The proof is by induction on the structure of M. The only possible situation is when both the surface reductions $M \to M_1', M_2'$ and $M \to M''$ are applied in surface contexts $\mathcal{C} \neq [\cdot]$, and we proceed by case analysis. We just consider a possible case. Suppose $M = PQ \to P_1'Q, P_2'Q$, where $P_1'Q = M_1'$ and $P_2'Q = M_2'$. Then either $M'' = P''Q$, where $P \to P''$, or $M'' = PQ''$, where $Q \to Q''$. In the first case we apply the induction hypothesis on $P \to P_1', P_2'$ and $P \to P''$ and we get that there exist R_1 and R_2 such that $P_1' \to R_1$, $P_2' \to R_2$ and $P'' \to R_1, R_2$, so that $P_1'Q \to R_1Q$, $P_2'Q \to R_2Q$ and $P''Q \to R_1Q, R_2Q$. In the second case, we have $P_1'Q \to P_1'Q''$, $P_2'Q \to P_2'Q''$ and $PQ'' \to P_1'Q'', P_2'Q''$. □

Lemma 60. *Let $M \in \Lambda^{!}_{l,\oplus}$. If $M \to M_1', M_2'$ and $M \to M_1'', M_2''$, with M_1', M_2', M_1'', M_2'' all distinct, then there exist N_1, N_2, N_3, N_4 such that $M_1' \to N_1, N_2$, $M_2' \to N_3, N_4$ and $\exists i \in \{1, 2\}$ such that $M_i'' \to N_1, N_3$ and $M_{3-i}'' \to N_2, N_4$.*

Proof. The proof is by induction on the structure of M. The only possible situation is when both the surface reductions $M \to M_1', M_2'$ and $M \to M_1'', M_2''$ are applied in surface contexts $\mathcal{C} \neq [\cdot]$, and we proceed by case analysis. We just consider a possible case. Suppose $M = PQ \to P_1'Q, P_2'Q$, where $M_1' = P_1'Q$ and $M_2' = P_2'Q$. Then either $M_1'' = P_1''Q$, $M_2'' = P_2''Q$ or $M_1'' = PQ_1''$, $M_2'' = PQ_2''$. In the first case we apply the induction hypothesis on $P \to P_1', P_2'$ and $P \to P_1'', P_2''$ and we have that there exist R_1, R_2, R_3, R_4 such that $P_1' \to R_1, R_2$, $P_2' \to R_3, R_4$ and $\exists i$ such that $P_i'' \to R_1, R_3$ and $P_{3-i}'' \to R_2, R_4$. Then, we have $P_1'Q \to R_1Q, R_2Q$, $P_2'Q \to R_3Q, R_4Q$, $P_i''Q \to R_1Q, R_3Q$, and $P_{3-i}''Q \to R_2Q, R_4Q$. In the second case we have $P_1'Q \to P_1'Q_1'', P_1'Q_2''$, $P_2'Q \to P_2'Q_1'', P_2'Q_2''$, $PQ_1'' \to P_1'Q_1'', P_1'Q_2''$, and $PQ_2'' \to P_1'Q_2'', P_2'Q_2''$. □

It is not trivial to prove the confluence of $\Rightarrow$. Following Dal Lago and Toldin [28], we first define a "laxer" probabilistic multi-step reduction relation $\Rrightarrow$ (Definition 50), i.e. such that $\Rightarrow \subseteq \Rrightarrow$, and then we prove two technical results on this relation (Lemma 62 and Lemma 63), from which we get the confluence of $\Rightarrow$.

Definition 50 (Multi-step reduction $\Rrightarrow$)**.**

- A *term distribution* is a probability distribution over $\Lambda^{!}_{\oplus}$, i.e. a function $\mathscr{D} : \Lambda^{!}_{\oplus} \longrightarrow [0, 1]$ such that:

$$\sum_{M \in \Lambda^{!}_{\oplus}} \mathscr{D}(M) = 1.$$

$$\frac{M \in \Lambda^!_\oplus}{M \Rrightarrow M}\ t1 \qquad\qquad \frac{M \to M_1, M_2 \qquad M_1 \Rrightarrow \mathscr{D}_1 \qquad M_2 \Rrightarrow \mathscr{D}_2}{M \Rrightarrow \dfrac{1}{2}\cdot \mathscr{D}_1 + \dfrac{1}{2}\cdot \mathscr{D}_2}\ t2$$

Figure 4.12: Multi-step reduction $\Rrightarrow$ for $\Lambda^!_\oplus$.

- The *multi-step reduction* $\Rrightarrow$ is the relation between terms of $\Lambda^!_\oplus$ and term distributions, defined by the rules in Figure 4.12.
- We inductively define the *size* $|\pi|$ of a derivation $\pi : M \Rrightarrow \mathscr{D}$ as follows:
  - if the last rule of π is $t1$ then $|\pi| \triangleq 0$;
  - otherwise, if the last rule of π is $t2$:

    $$\frac{M \to M_1, M_2 \qquad \pi_1 : M_1 \Rrightarrow \mathscr{D}_1 \qquad \pi_2 : M_2 \Rrightarrow \mathscr{D}_2}{M \Rrightarrow \frac{1}{2}\cdot \mathscr{D}_1 + \frac{1}{2}\cdot \mathscr{D}_2}\ t2$$

    then $|\pi| \triangleq \max(|\pi_1|, |\pi_2|) + 1$.

  Henceforth, with a little abuse of notation, we shall write $|M \Rrightarrow \mathscr{D}|$ in place of $|\pi|$, whenever $\pi : M \Rrightarrow \mathscr{D}$.

As in the case of $\Rightarrow$, we apply the basic definitions and conventions related to probability distributions and relations of Section 2.5.2.

Notice that the only difference between the relations $\Rightarrow$ and $\Rrightarrow$ is that $s1$ applies to surface normal forms only, while $t1$ applies to all terms. The following states that $\Rightarrow \subset \Rrightarrow$:

Lemma 61. *If $\pi : M \Rightarrow \mathscr{D}$ then there exists a derivation π' such that $\pi' : M \Rrightarrow \mathscr{D}$ and $|\pi| = |\pi'|$.*

The confluence of "$\Rrightarrow$" requires two technical lemmas, following [28]:

Lemma 62. *Let $M \in \Lambda^!_{l,\oplus}$. Let $M \Rrightarrow \mathscr{D}$ be such that $\mathscr{D} = p_1 \cdot N_1 + \ldots + p_n \cdot N_n$, and let $N_i \Rrightarrow \mathscr{E}_i$ for all $i \leq n$. Then:*

(1) $M \Rrightarrow \sum_{i=1}^n p_i \cdot \mathscr{E}_i$

(2) $|M \Rrightarrow \sum_{i=1}^n p_i \cdot \mathscr{E}_i| \leq |M \Rrightarrow \mathscr{D}| + \max_{i=1}^n |N_i \Rrightarrow \mathscr{E}_i|$.

Proof. By induction on the structure of the derivation of $M \Rrightarrow \mathscr{D}$. If the last rule is $t1$ then $\mathscr{D} = M$ and there is nothing to prove. If the last rule is $t2$ then $M \Rrightarrow \mathscr{D}$ as follows:

$$\frac{M \to M_1, M_2 \qquad M_1 \Rrightarrow \mathscr{D}_1 \qquad M_2 \Rrightarrow \mathscr{D}_2}{M \Rrightarrow \frac{1}{2}\cdot \mathscr{D}_1 + \frac{1}{2}\cdot \mathscr{D}_2}\ t2$$

We have two cases depending on the first premise. If $M_1 = M_2$ then by induction hypothesis on $M_1 \Rrightarrow \mathscr{D}_1$ there exists $M_1 \Rrightarrow \sum_{i=1}^n p_i \cdot \mathscr{E}_i$. By applying the rule $t2$ we get $M \Rrightarrow \sum_{i=1}^n p_i \cdot \mathscr{E}_i$. Otherwise, $M_1 \neq M_2$. W.l.o.g. let us assume that $\mathscr{D}_1 = 2p_1 \cdot N_1 + \ldots + 2p_{o-1} \cdot N_{o-1} + p_o \cdot N_o + \ldots + p_m \cdot N_m$ and $\mathscr{D}_2 = p_o \cdot N_o + \ldots + p_m \cdot N_m + 2p_{m+1} \cdot N_{m+1} + \ldots + 2p_n \cdot N_n$, where $1 \leq o \leq m \leq n$. By applying the induction hypothesis on $M_1 \Rrightarrow \mathscr{D}_1$ and $M_2 \Rrightarrow \mathscr{D}_2$ we have that there exist $M_1 \Rrightarrow \mathscr{P}_1$

and $M_2 \Rrightarrow \mathscr{P}_2$, where $\mathscr{P}_1 = \sum_{i=1}^{o-1} 2p_i \cdot \mathscr{E}_i + \sum_{i=o}^{m} p_i \cdot \mathscr{E}_i$ and $\mathscr{P}_2 = \sum_{i=o}^{m} p_i \cdot \mathscr{E}_i + \sum_{i=m+1}^{n} 2p_i \cdot \mathscr{E}_i$. By applying rule $t2$ we have $M \Rrightarrow \sum_{i=1}^{n} p_i \cdot \mathscr{E}_i$. Concerning the bound on the derivation, the induction hypothesis on $M_1 \Rrightarrow \mathscr{D}_1$ and $M_2 \Rrightarrow \mathscr{D}_2$ gives $|M_1 \Rrightarrow \mathscr{P}_1| \le |M_1 \Rrightarrow \mathscr{D}_1| + \max_{i=1}^{m} |N_i \Rrightarrow \mathscr{E}_i|$ and $|M_2 \Rrightarrow \mathscr{P}_2| \le |M_1 \Rrightarrow \mathscr{D}_1| + \max_{i=o}^{n} |N_i \Rrightarrow \mathscr{E}_i|$. Then, we have:

$$
\begin{aligned}
&|M \Rrightarrow \sum_{i=1}^{n} p_i \cdot \mathscr{E}_i| \\
&= \max(|M_1 \Rrightarrow \mathscr{P}_1|, |M_2 \Rrightarrow \mathscr{P}_2|) + 1 \\
&\le \max((|M_1 \Rrightarrow \mathscr{D}_1| + \max_{i=1}^{m} |N_i \Rrightarrow \mathscr{E}_i|), (|M_2 \Rrightarrow \mathscr{D}_2| + \max_{i=o}^{n} |N_i \Rrightarrow \mathscr{E}_i|)) + 1 \\
&\le \max(|M_1 \Rrightarrow \mathscr{D}_1|, |M_2 \Rrightarrow \mathscr{D}_2|) + \max((\max_{i=1}^{m} |N_i \Rrightarrow \mathscr{E}_i|), (\max_{i=o}^{n} |N_i \Rrightarrow \mathscr{E}_i|))) + 1 \\
&\le |M \Rrightarrow \mathscr{D}| + \max_{i=1}^{n} |N_i \Rrightarrow \mathscr{E}_i|
\end{aligned}
$$

This concludes the proof. $\square$

Lemma 63. *Let $M \in \Lambda^{!}_{l,\oplus}$. If $M \Rrightarrow \mathscr{D}$ and $M \Rrightarrow \mathscr{E}$, where $\mathscr{D} = p_1 \cdot P_1 + \ldots + p_n \cdot P_n$ and $\mathscr{E} = q_1 \cdot Q_1 + \ldots + q_m \cdot Q_m$, then there exist $\mathscr{L}_1, \ldots \mathscr{L}_n$ and $\mathscr{F}_1 \ldots \mathscr{F}_m$ such that:*

- *$P_i \Rrightarrow \mathscr{L}_i$ and $Q_j \Rrightarrow \mathscr{F}_j$, for all $i \le n$, $j \le m$;*
- *$\max_{i=1}^{n} |P_i \Rrightarrow \mathscr{L}_i| \le |M \Rrightarrow \mathscr{E}|$ and $\max_{j=1}^{m} |Q_j \Rrightarrow \mathscr{F}_j| \le |M \Rrightarrow \mathscr{D}|$;*
- *$\sum_{i=1}^{n} p_i \cdot \mathscr{L}_i = \sum_{j=1}^{m} q_j \cdot \mathscr{F}_j$.*

Proof. By induction on $|M \Rrightarrow \mathscr{D}| + |M \Rrightarrow \mathscr{E}|$. If both $M \Rrightarrow \mathscr{D}$ and $M \Rrightarrow \mathscr{E}$ end with $t1$ then there is nothing to prove. If only one of the derivations end with $t1$ then we are in the following case:

$$
\frac{M \to M_1, M_2 \qquad M_1 \Rrightarrow \mathscr{D}_1 \qquad M_2 \Rrightarrow \mathscr{D}_2}{M \Rrightarrow \frac{1}{2} \cdot \mathscr{D}_1 + \frac{1}{2} \cdot \mathscr{D}_2}\, t2 \qquad \frac{}{M \Rrightarrow M}\, t1
$$

and the result follows. Otherwise, both derivations $M \Rrightarrow \mathscr{D}$ and $M \Rrightarrow \mathscr{E}$ end with the rule $t2$

$$
\frac{M \to M_1, M_2 \qquad M_1 \Rrightarrow \mathscr{D}_1 \qquad M_2 \Rrightarrow \mathscr{D}_2}{M \Rrightarrow \frac{1}{2} \cdot \mathscr{D}_1 + \frac{1}{2} \cdot \mathscr{D}_2}\, t2 \qquad \frac{M \to N_1, N_2 \qquad N_1 \Rrightarrow \mathscr{E}_1 \qquad N_2 \Rrightarrow \mathscr{E}_2}{M \Rrightarrow \frac{1}{2} \cdot \mathscr{E}_1 + \frac{1}{2} \cdot \mathscr{E}_2}\, t2
$$

Clearly, if M_1, M_2 is equal to N_1, N_2 (modulo sort) then we apply the induction hypothesis and we are done. So let us suppose that M_1, M_2 and N_1, N_2 are different. We have four cases:

- If $M_1 = M_2$ and $N_1 = N_2$ then by Lemma 58 there exists L such that $M_1 \to L$ and $N_1 \to L$. By using the rule $t1$ we get $L \Rrightarrow L$. By induction hypothesis on $M_1 \Rrightarrow \mathscr{D}_1$ and $L \Rrightarrow L$ there exist $\mathscr{L}_1, \ldots, \mathscr{L}_n$ and $\mathscr{K}$ such that, for all $i \le n$, $P_i \Rrightarrow \mathscr{L}_i$, $L \Rrightarrow \mathscr{K}$, $\max_{i=1}^{n}(|P_i \Rrightarrow \mathscr{L}_i|) \le |M \Rrightarrow L|$, $|L \Rrightarrow \mathscr{K}| \le |M \Rrightarrow \mathscr{D}|$, and $\sum_{i=1}^{n} p_i \cdot \mathscr{L}_i = \mathscr{K}$. Similarly, we have that there exists $\mathscr{F}_1, \ldots, \mathscr{F}_m, \mathscr{H}$ such that, for all $i \le m$, $Q_i \Rrightarrow \mathscr{F}_i$, $L \Rrightarrow \mathscr{H}$, $\max_{i=1}^{m}(|Q_i \Rrightarrow \mathscr{F}_i|) \le |M \Rrightarrow L|$, $|L \Rrightarrow \mathscr{H}| \le |M \Rrightarrow \mathscr{E}|$, and $\sum_{i=1}^{m} q_i \cdot \mathscr{F}_i = \mathscr{H}$. We obtain $|L \Rrightarrow \mathscr{K}| + |L \Rrightarrow \mathscr{H}| \le |M \Rrightarrow \mathscr{D}| + |M \Rrightarrow \mathscr{E}|$. Let $\mathscr{K} = r_1 \cdot R_1 + \ldots + r_h \cdot R_h$ and $\mathscr{H} = s_1 \cdot S_1 + \ldots + s_k \cdot S_k$. We apply the induction hypothesis and we obtain that there exist $\mathscr{R}_1, \ldots, \mathscr{R}_h, \mathscr{S}_1, \ldots, \mathscr{S}_k$ such that $R_i \Rrightarrow \mathscr{R}_i$ and $S_j \Rrightarrow \mathscr{S}_j$ for all $i \le h$ and $j \le k$. Moreover, $\max_{i=1}^{h}(|R_i \Rrightarrow \mathscr{R}_i|) \le |L \Rrightarrow \mathscr{H}|$, $\max_{k=1}^{k}(|S_j \Rrightarrow \mathscr{S}_j|) \le |L \Rrightarrow \mathscr{K}|$, and $\sum_{i=1}^{h} r_i \cdot \mathscr{R}_i = \sum_{j=1}^{k} s_j \cdot \mathscr{S}_j$.Notice that the cardinality of $\mathscr{D}$ and $\mathscr{K}$ may differ but for sure they have the same terms with non zero probability. Similar, $\mathscr{E}$ and $\mathscr{H}$ have the same terms with non zero probability. By using Lemma 62 and using the transitive property of equality we obtain that $\sum_{i=1}^{n} p_i \cdot \mathscr{R}_i = \sum_{i=1}^{n} r_i \cdot \mathscr{R}_i = \sum_{j=1}^{m} s_j \cdot \mathscr{S}_j = \sum_{j=1}^{m} q_j \cdot \mathscr{S}_j$. Moreover, we have

$$\max_{i=1}^{n}(|P_i \Rrightarrow \mathscr{R}_i|) \leq |L \Rrightarrow \mathscr{H}| \leq |M \Rrightarrow \mathscr{E}|$$
$$\max_{j=1}^{m}(|Q_j \Rrightarrow \mathscr{S}_j|) \leq |L \Rrightarrow \mathscr{K}| \leq |M \Rrightarrow \mathscr{D}|.$$

- If $M_1 \neq M_2$ and $N_1 = N_2$ then by Lemma 59 there exists L_1, L_2 such that $M_1 \to L_1$, $M_2 \to L_2$ and $N_1 \to L_1, L_2$. W.l.o.g. we can assume that $\mathscr{D}_1 = 2p_1 \cdot P_1 + \ldots + 2p_{o-1} \cdot P_{o-1} + p_o \cdot P_o + \ldots + p_t \cdot P_t$ and $\mathscr{D}_2 = p_o \cdot P_o + \ldots + p_h \cdot P_h + 2p_{t+1} \cdot P_{t+1} + \ldots + 2p_n \cdot P_n$ where $1 \leq o \leq t \leq n$.By using the induction rule, we associate with every L_i a distribution $\mathscr{P}_i$ such that $L_1 \Rrightarrow \mathscr{P}_1$ and $L_2 \Rrightarrow \mathscr{P}_2$. Let $\mathscr{P}_1 = r_1 \cdot R_1 + \ldots + r_h \cdot R_h$ and $\mathscr{P}_2 = s_1 \cdot S_1 + \ldots + s_k \cdot S_k$. So, we have, for all i, $M_i \Rrightarrow \mathscr{D}_i$ and $M_i \Rrightarrow \mathscr{P}_i$, $N_1 \Rrightarrow \mathscr{E}$ and $N_1 \Rrightarrow \frac{1}{2} \cdot \mathscr{P}_1 + \frac{1}{2} \cdot \mathscr{P}_2$. By applying the induction hypothesis on all the three cases we have that there exist $\mathscr{L}_1, \ldots, \mathscr{L}_n, \mathscr{F}_1, \ldots, \mathscr{F}_m, \mathscr{K}, \mathscr{H}, \mathscr{R}, \mathscr{S}$ such that $P_1 \Rrightarrow \mathscr{L}_1, \ldots, P_n \Rrightarrow \mathscr{L}_n$, $Q_1 \Rrightarrow \mathscr{F}_1$, …, $P_m \Rrightarrow \mathscr{F}_m$, $L_1 \Rrightarrow \mathscr{K}$, $L_2 \Rrightarrow \mathscr{H}$, $L_1 \Rrightarrow \mathscr{R}$, and $L_2 \Rrightarrow \mathscr{S}$. Moreover:

  1. $\max_{1 \leq i \leq t}(|P_i \Rrightarrow \mathscr{L}_i|) \leq |M_1 \Rrightarrow \mathscr{P}_1|$, $|L_1 \Rrightarrow \mathscr{K}| \leq |M_1 \Rrightarrow \mathscr{D}_1|$, and $\sum_{i=1}^{o-1} 2p_i \cdot \mathscr{L}_i + \sum_{i=o}^{t} p_i \cdot \mathscr{L}_i = \mathscr{K}$.
  2. $\max_{o \leq i \leq n}(|P_1 \Rrightarrow \mathscr{L}_i|) \leq |M_2 \Rrightarrow \mathscr{P}_2|$, $|L_2 \Rrightarrow \mathscr{H}| \leq |M_2 \Rrightarrow \mathscr{D}_2|$, and $\sum_{i=o}^{t} p_i \cdot \mathscr{L}_i + \sum_{i=t+1}^{n} 2p_i \cdot \mathscr{L}_i = \mathscr{H}$.
  3. $\max_{j=1}^{m}(|Q_j \Rrightarrow \mathscr{F}_j|) \leq |N_1 \Rrightarrow \frac{1}{2} \cdot \mathscr{P}_1 + \frac{1}{2} \cdot \mathscr{P}_2|$, $\max(|L_1 \Rrightarrow \mathscr{R}|, |L_2 \Rrightarrow \mathscr{S}|) \leq |N_1 \Rrightarrow \mathscr{E}|$, and $\sum_{j=1}^{m} q_j \cdot \mathscr{F}_j = \frac{1}{2} \cdot \mathscr{R} + \frac{1}{2} \cdot \mathscr{S}$.

  Notice that $|L_1 \Rrightarrow \mathscr{R}| + |L_1 \Rrightarrow \mathscr{K}| < |M \Rrightarrow \mathscr{D}| + |M \Rrightarrow \mathscr{E}|$. Moreover, notice also that the following inequality holds: $|L_2 \Rrightarrow \mathscr{S}| + |L_2 \Rrightarrow \mathscr{H}| < |M \Rrightarrow \mathscr{D}| + |M \Rrightarrow \mathscr{E}|$. We are allowed to apply, again, induction hypothesis and have a confluent distribution for both cases. Lemma 62 then allows us to connect the first two main derivations and by transitivive property of equality we have the thesis.

- The case $M_1 = M_2$ and $N_1 \neq N_2$ and the case $M_1 \neq M_2$ and $N_1 \neq N_2$ are similar by using, respectively, Lemma 59 and Lemma 60. □

Theorem 64 (Confluence for $\Rightarrow$)**.** *Let $M \in \Lambda^{!}_{l,\oplus}$. If $M \Rightarrow \mathscr{D}$ and $M \Rightarrow \mathscr{E}$ then $\mathscr{D} = \mathscr{E}$.*

Proof. Since $\Rightarrow \subseteq \Rrightarrow$, we have $M \Rrightarrow \mathscr{D}$ and $M \Rrightarrow \mathscr{E}$. By Lemma 63, $\mathscr{D} = \mathscr{E}$. □

4.3.2 Weighted subject reduction

In the previous subsection we have shown that for each term $M \in \Lambda^{!}_{l,\oplus}$ there exists *at most one* surface distribution $\mathscr{D}$ such that $M \Rightarrow \mathscr{D}$ (Theorem 64), so the relation $\Rightarrow$ is always well-defined in $\Lambda^{!}_{l,\oplus}$. However, not all terms of $\Lambda^{!}_{l,\oplus}$ can be put in relation with a surface distribution, as the following example shows.

Example 20. Consider the terms $\mathbf{\Delta}_! \triangleq \lambda !x.\mathtt{d}(x)!\mathtt{d}(x)$ and $\mathbf{\Omega}_! \triangleq \mathbf{\Delta}_!(!\mathbf{\Delta}_!)$ in $\Lambda^{!}_{l,\oplus}$. By applying a surface reduction step to $\mathbf{\Omega}_!$ we have:

$$\begin{aligned}\mathbf{\Omega}_! &\to (\mathtt{d}(x)!\mathtt{d}(x))\{!\mathbf{\Delta}_!/x\} \\ &= ((y\,!y)[\mathtt{d}(x)/y])\{!\mathbf{\Delta}_!/x\} = \mathbf{\Omega}_!\end{aligned}$$

hence $\mathbf{\Omega}_!$ reduces to itself.

We now show that, as long as we restrict to terms M typable in $\mathsf{STA}_\oplus$, there always exists at least one surface distribution $\mathscr{D}$ such that $M \Rrightarrow \mathscr{D}$ (Theorem 72): we first associate with each term of $\Lambda^!_{l,\oplus}$ a "weight", and then we prove that reducing a typable term yields a typable term with strictly smaller weight. Actually, this result is nothing more than a stronger version of the Subject reduction property.

To begin with, we introduce some basic definitions from [56, 40], together with the new notion of "co-rank". The latter plays a role in the definition of weight, and is due to the presence of explicit constructs ! and $\mathtt{d}$, which affect the size of a term.

Definition 51 (Rank, co-rank, weight, depth)**.**

- The *rank* of a rule m of the form:

$$\frac{\Gamma, x_1 : \sigma, \ldots, x_n : \sigma \vdash M : \tau \qquad (n \geq 0)}{\Gamma, x : !\sigma \vdash M[\mathtt{d}(x)/x_1, \ldots, \mathtt{d}(x)/x_n] : \tau}\, m$$

  is the number $k \leq n$ of variables x_i such that $x_i \in \mathrm{FV}(M)$. If $\mathcal{D}$ is a derivation, the *rank of* $\mathcal{D}$, written $\mathrm{rk}(\mathcal{D})$, is $\max(1, k)$, where k is the maximum rank among the applications of the rule m in $\mathcal{D}$.

- The *co-rank of a rule sp* of the form:

$$\frac{x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau}{y_1 : !\sigma_1, \ldots, y_n : !\sigma_n \vdash !M[\mathtt{d}(y_1)/x_1, \ldots, \mathtt{d}(y_n)/y_n] : \tau}\, sp$$

  is the number $k \leq n$ of variables x_i such that $x_i \in \mathrm{FV}(M)$.

- Let $n \geq 1$. The *weight* of a derivation $\mathcal{D}$ with respect to n, written $\mathrm{w}(\mathcal{D}, n)$, is defined by induction on the structure of $\mathcal{D}$ as follows:
  - if the last rule is ax then $\mathrm{w}(\mathcal{D}, n) = 1$;
  - if the last rule is $\multimap$R or $\wedge$E with premise a derivation $\mathcal{D}'$ then $\mathrm{w}(\mathcal{D}, n) = \mathrm{w}(\mathcal{D}', n) + 1$;
  - if the last rule is $\multimap$E or $\wedge$I0 with premises $\mathcal{D}'$ and $\mathcal{D}''$ then $\mathrm{w}(\mathcal{D}, n) = \mathrm{w}(\mathcal{D}', n) + \mathrm{w}(\mathcal{D}'', n) + 1$;
  - if the last rule is $\wedge$I1 with premises, respectively, $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$, $\mathcal{D}_4$, then $\mathrm{w}(\mathcal{D}, n) = \mathrm{w}(\mathcal{D}_1, n) + \mathrm{w}(\mathcal{D}_2, n) + \mathrm{w}(\mathcal{D}_3, n) + \mathrm{w}(\mathcal{D}_4, n) + 2$;
  - if the last rule is $\forall$I, or $\forall$E with premise $\mathcal{D}'$ then $\mathrm{w}(\mathcal{D}, n) = \mathrm{w}(\mathcal{D}', n)$;
  - if the last rule is sp with premise $\mathcal{D}'$ and co-rank k then $\mathrm{w}(\mathcal{D}, n) = n \cdot (\mathrm{w}(\mathcal{D}', n) + k) + 1$;
  - if the last rule is m with premise $\mathcal{D}'$ and rank k then $\mathrm{w}(\mathcal{D}, n) = \mathrm{w}(\mathcal{D}', n) + k$.

- The *depth* of a derivation $\mathcal{D}$, denoted $\mathrm{d}(\mathcal{D})$, is the maximal number of nesting of applications of the sp rule in $\mathcal{D}$, i.e. the maximal number of applications of the sp rule in a path connecting the conclusion and one axiom in $\mathcal{D}$.

Following [40], we state and prove the basic properties relating rank, size and weight.

Lemma 65. *Let $n \geq 1$, and let $\mathcal{D} \triangleleft \Gamma \vdash_{\mathsf{STA}_\oplus} M : \sigma$. Then:*

(1) $\mathrm{rk}(\mathcal{D}) \leq |M|$;

(2) $\mathrm{w}(\mathcal{D}, n) \leq n^{\mathrm{d}(\mathcal{D})} \cdot \mathrm{w}(\mathcal{D}, 1)$;

(3) $\mathrm{w}(\mathcal{D},1)=|M|$. *If, moreover, $\mathcal{D}$ is free from applications of sp and m, then* $\mathrm{w}(\mathcal{D},n)=|M|$.

Proof. By induction on the structure of $\mathcal{D}$. Point (1) and point (3) are straightforward. Concerning point (2), we consider the most interesting case in which $\mathcal{D}$ has been obtained from a derivation $\mathcal{D}'$ by applying the rule *sp* with co-rank k. By using the induction hypothesis, we have:

$$\begin{aligned}\mathrm{w}(\mathcal{D},n)&=n\cdot(\mathrm{w}(\mathcal{D}',n)+k)+1\\&\leq n\cdot(n^{\mathrm{d}(\mathcal{D}')}\cdot\mathrm{w}(\mathcal{D}',1)+k)+1\\&\leq n\cdot(n^{\mathrm{d}(\mathcal{D}')}\cdot\mathrm{w}(\mathcal{D}',1)+n^{\mathrm{d}(\mathcal{D}')}\cdot k)+n^{\mathrm{d}(\mathcal{D}')+1}\\&\leq n^{\mathrm{d}(\mathcal{D}')+1}\cdot(\mathrm{w}(\mathcal{D}',1)+k+1)=n^{\mathrm{d}(\mathcal{D})}\cdot\mathrm{w}(\mathcal{D},1).\end{aligned}$$

□

The Subject reduction property requires some technical, standard results. First, we notice that substituting some applications of the rule *ax* with others applications of the same rule does not affect the weight of the derivation:

Lemma 66. *For every type A, if $\mathcal{D}\lhd\Gamma\vdash M:\tau$ then there exits $\mathcal{D}'$ such that $\mathcal{D}'\lhd\Gamma[A/\alpha]\vdash M:\tau[A/\alpha]$ and $\mathrm{w}(\mathcal{D},n)=\mathrm{w}(\mathcal{D}',n)$.*

In analogy with [40], so relying on the structure of the essential types, by inspecting the rules of $\mathsf{STA}_\oplus$ it holds:

Lemma 67.

(1) If $\mathcal{D}\lhd\Gamma\vdash M:\,!\sigma$ then $\mathcal{D}$ has been obtained from a derivation $\mathcal{D}'$ by applying the rule sp, followed by some applications of the rule m. Hence, Γ is a strictly exponential context and $M=\,!M'$, for some M'.

(2) If $\mathcal{D}\lhd\Gamma,x:\,!\sigma\vdash M:\tau$. Then, either $x:\,!\sigma$ has been introduced by a sp rule or by a m rule.

Lemma 68 (Generation)**.**

(1) If $\mathcal{D}\lhd\Gamma\vdash x:\sigma$ then $\sigma=\forall\vec{\alpha}.(B\langle D_1/\beta_1,\dots,D_n/\beta_n\rangle)$ and $\mathcal{D}$ is an instance of ax with conclusion $x:B\vdash x:B$ followed by a sequence of $\forall\mathrm{I}$ and $\forall\mathrm{E}$, where $\vec{\alpha}=\alpha_1,\dots,\alpha_k$, for some $k\geq 0$.

(2) If $\mathcal{D}\lhd\Gamma\vdash\lambda(!)x.M:\sigma$ then $\sigma=\forall\vec{\alpha}.((\tau\multimap A)\langle D_1/\beta_1,\dots,D_n/\beta_n\rangle)$ and $\mathcal{D}$ is some $\mathcal{D}'\lhd\Gamma,x:\tau\vdash M':A$ followed by $\multimap\mathrm{I}$ and a sequence of $\forall\mathrm{I}$, $\forall\mathrm{E}$, and m where $\vec{\alpha}=\alpha_1,\dots,\alpha_k$, for some $k\geq 0$.

(3) If $\mathcal{D}\lhd\Gamma\vdash MN:\sigma$ then $\sigma=\forall\vec{\alpha}.(A\langle D_1/\beta_1,\dots,D_n/\beta_n\rangle)$ and $\mathcal{D}$ is some $\mathcal{D}'\lhd\Gamma'\vdash M':\tau\multimap A$ and $\mathcal{D}''\lhd\Gamma''\vdash N':\tau$ followed by $\multimap\mathrm{E}$ and a sequence of $\forall\mathrm{I}$, $\forall\mathrm{E}$, and m, where $\vec{\alpha}=\alpha_1,\dots,\alpha_k$ and $k\geq 0$.

(4) If $\mathcal{D}\lhd\Gamma\vdash\mathtt{copy}^W_A\,N\ \mathtt{as}\ x_1,x_2\ \mathtt{in}\ (M_1,M_2):\sigma$, then $\sigma=B_1\wedge B_2$ and $\mathcal{D}$ is $\wedge\mathrm{I}1$ followed by a sequence of applications of the rule m.

(5) If $\mathcal{D}\lhd\Gamma\vdash(M_1,M_2):\sigma$ then $\sigma=B_1\wedge B_2$ and $\mathcal{D}$ is $\mathcal{D}'\lhd\vdash M_1':B_1$ and $\mathcal{D}''\lhd\vdash M_2':B_2$ followed by $\wedge\mathrm{I}0$ and a sequence of m with rank 0 introducing the context Γ.

(6) If $\mathcal{D}\lhd\Gamma\vdash\mathtt{proj}^{B_1\wedge B_2}_{B_i}(M):\sigma$ then $\sigma=\forall\vec{\alpha}.(B_i\langle D_1/\beta_1,\dots,D_n/\beta_n\rangle)$, and $\mathcal{D}$ is $\mathcal{D}'\lhd\Gamma'\vdash M':B_1\wedge B_2$ followed by $\wedge\mathrm{E}$ and a sequence of $\forall\mathrm{I}$, $\forall\mathrm{E}$, and m, where $\vec{\alpha}=\alpha_1,\dots,\alpha_k$, for some $k\geq 0$.

(7) *If $\mathcal{D} \lhd \Gamma \vdash !M : \sigma$ then $\sigma = !\sigma'$, Γ is an strictly exponential context, and $\mathcal{D}$ is sp, followed by some applications of the rule m.*

Proof. Points (1)-(6) are an easy generalization of Lemma 44. Point (7) is straightforward. □

Following Gaboardi and Ronchi [40], we prove a "weighted" formulation of the substitution property. Since we work with two kinds of types, namely the linear and the strictly exponential ones, we split the task. First we consider a substitution theorem for to linear types, i.e. those with form A. Then we generalize the statement to arbitrary types, i.e. those with form σ.

Lemma 69. *If $\mathcal{D} \lhd \Gamma, x : A \vdash M : \tau$ then x is s-linear in M.*

Proof. Straightforward. □

Lemma 70 (Weighted linear substitution)**.** *Let $n \geq 1$. If $\mathcal{D}_1 \lhd \Gamma, x : A \vdash M : \tau$ and $\mathcal{D}_2 \lhd \Delta \vdash N : A$, then there exists a derivation $S(\mathcal{D}_1, \mathcal{D}_2)$ such that:*

- $S(\mathcal{D}_1, \mathcal{D}_2) \lhd \Gamma, \Delta \vdash M[N/x] : \tau$,
- $\mathrm{w}(S(\mathcal{D}_1, \mathcal{D}_2), n) \leq \mathrm{w}(\mathcal{D}_1, n) + \mathrm{w}(\mathcal{D}_2, n)$.

Proof. By Lemma 69, x is s-linear in M, i.e. x occurs exactly once in M and this occurrence is out of the scope of both a !-operator and a $\mathtt{d}$-operator. The statement is proved by induction on $\mathcal{D}_1$. The cases were the last rule is ax, $\multimap$I, $\multimap$E, $\wedge$I0, $\wedge$I1, $\forall$I, $\forall$E, and m are easy. Now, suppose $\mathcal{D}_1$ is of the form:

$$\frac{\mathcal{D}' \lhd \Gamma, x : A \vdash P : B \quad \mathcal{D}'' \lhd x_1 : B \vdash Q_1 : C_1 \quad \mathcal{D}''' \lhd x_2 : B \vdash Q_2 : C_2 \quad \mathcal{D}'''' \lhd \vdash W : B}{\Gamma, x : A \vdash \mathtt{copy}^W_B\ P\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (Q_1, Q_2) : C_1 \wedge C_2}\wedge\mathrm{I1}$$

so that $\tau = C_1 \wedge C_2$ and $M = \mathtt{copy}^W_B\ P\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (Q_1, Q_2)$. By induction hypothesis, there exists $S(\mathcal{D}', \mathcal{D}_2) \lhd \Gamma \vdash P[N/x] : A$ such that $\mathrm{w}(S(\mathcal{D}', \mathcal{D}_2), n) \leq \mathrm{w}(\mathcal{D}', n) + \mathrm{w}(\mathcal{D}_2, n)$. We define $S(\mathcal{D}_1, \mathcal{D}_2) \lhd \Gamma, \Delta \vdash \mathtt{copy}^W_B\ P[N/x]\ \mathtt{as}\ x_1, x_2\ \mathtt{in}\ (Q_1, Q_2) : C_1 \wedge C_2$ as the derivation obtained by applying $\wedge$I1 to $S(\mathcal{D}', \mathcal{D}_2)$, $\mathcal{D}''$, $\mathcal{D}'''$, $\mathcal{D}''''$. Moreover, by using the induction hypothesis, we have:

$$\begin{aligned}\mathrm{w}(S(\mathcal{D}_1, \mathcal{D}_2), n) &= \mathrm{w}(S(\mathcal{D}', \mathcal{D}_2), n) + \mathrm{w}(\mathcal{D}'', n) + \mathrm{w}(\mathcal{D}''', n) + \mathrm{w}(\mathcal{D}'''', n) + 2 \\ &\leq \mathrm{w}(\mathcal{D}', n) + \mathrm{w}(\mathcal{D}'', n) + \mathrm{w}(\mathcal{D}''', n) + \mathrm{w}(\mathcal{D}'''', n) + \mathrm{w}(\mathcal{D}_2, n) + 2 \\ &= \mathrm{w}(\mathcal{D}_1, n) + \mathrm{w}(\mathcal{D}_2, n).\end{aligned}$$

Last, since A is a linear type, the last rule of $\mathcal{D}_1$ cannot be sp. □

Lemma 71 (Weighted substitution)**.** *Let $r \geq \mathrm{rk}(\mathcal{D}_1)$. If $\mathcal{D}_1 \lhd \Gamma, x : \sigma \vdash M : \tau$ and $\mathcal{D}_2 \lhd \Delta \vdash N : \sigma$, then there exists a derivation $S(\mathcal{D}_1, \mathcal{D}_2)$ such that:*

- $S(\mathcal{D}_1, \mathcal{D}_2) \lhd \Gamma, \Delta \vdash M\{N/x\} : \tau$,
- $\mathrm{w}(S(\mathcal{D}_1, \mathcal{D}_2), r) \leq \mathrm{w}(\mathcal{D}_1, r) + \mathrm{w}(\mathcal{D}_2, r)$.

Proof. Since $\sigma = !^q A$, for some linear type A and some $q \geq 0$, we reason by induction on q. If $q = 0$ then, by Lemma 69, x is s-linear in M, i.e. x occurs exactly once in M and this occurrence is out of the scope of both a !-operator and a $\mathtt{d}$-operator. This means that $M\{N/x\} = M[N/x]$, and we can apply Lemma 70. Suppose now that $\sigma = !\sigma'$. On the one hand, by Lemma 67.(1) we have that Δ is strictly exponential, $N = !P$, and $\mathcal{D}_2$ is composed by a subderivation $\mathcal{D}_2^*$ of the form:

$$\dfrac{\begin{array}{c}\mathcal{D}'_2\\ \Delta' \vdash P' : \sigma'\end{array}}{!\Delta' \vdash !P'[\mathtt{d}(z_1)/y_1, \dots, \mathtt{d}(z_m)/y_m] : !\sigma'}\ sp$$

with co-rank h and such that $\Delta' = y_1 : \sigma_1, \dots, y_m : \sigma_m$, followed by a sequence of $t \geq 0$ rules m with rank, respectively, $k_1, \dots, k_t$ recovering $\Delta \vdash !P : !\sigma'$. On the other hand, by applying Lemma 67.(2), the assumption $x : !\sigma'$ in $\mathcal{D}_1 \lhd \Gamma, x : !\sigma' \vdash M : \tau$ has been obtained by applying in $\mathcal{D}$ either the rule *sp* or the rule m. We just consider the latter case, the former being similar. W.l.o.g. we can suppose that such an instance of m is the last rule of $\mathcal{D}_1$, since we can always permute an application of m downward obtaining a derivation of the same judgement. Then, $\mathcal{D}_1$ has the following form:

$$\dfrac{\begin{array}{c}\mathcal{D}'_1\\ \Gamma, x_1 : \sigma', \dots, x_n : \sigma' \vdash M' : \tau \qquad (n \geq 0)\end{array}}{\Gamma, x : !\sigma' \vdash M'[\mathtt{d}(x)/x_1, \dots, \mathtt{d}(x)/x_n] : \tau}\ m$$

with rank k and such that $M = M'[\mathtt{d}(x)/x_1, \dots, \mathtt{d}(x)/x_n]$. If $k = 0$ then $S(\mathcal{D}_1, \mathcal{D}_2)$ is $\mathcal{D}'_1$ followed by some applications of the m rule with rank 0 in order to recover the context Δ, which is strictly exponential by Lemma 67.(1). In this case, we have $\mathrm{w}(S(\mathcal{D}_1, \mathcal{D}_2), r) = \mathrm{w}(\mathcal{D}'_1, r)$. Otherwise, by using the induction hypothesis, we can build the following derivations:

$$\begin{aligned}
&S^1 \triangleq S(\mathcal{D}'_2, \mathcal{D}'_1) \lhd \Gamma, \Delta', x_2 : \sigma', \dots, x_n : \sigma' \vdash M'\{P'/x_1\} : \tau\\
&S^2 \triangleq S(\mathcal{D}'_2, S(\mathcal{D}'_2, \mathcal{D}'_1)) \lhd \Gamma, \Delta', \Delta', x_3 : \sigma', \dots, x_n : \sigma' \vdash M'\{P'/x_1, P'/x_2\} : \tau\\
&\quad \dots\\
&S^n \triangleq S(\mathcal{D}'_2, S(\mathcal{D}'_2, \dots S(\mathcal{D}'_2, \mathcal{D}'_1))) \lhd \Gamma, \Delta', \overset{n}{\dots}, \Delta' \vdash M'\{P'/x_1, \dots, P'/x_n\} : \tau.
\end{aligned}$$

such that $\mathrm{w}(S^1, r) \leq \mathrm{w}(\mathcal{D}'_2) + \mathrm{w}(\mathcal{D}'_1)$ and, for all $1 \leq i < n$, $\mathrm{w}(S^{i+1}, r) \leq \mathrm{w}(\mathcal{D}'_2, r) + \mathrm{w}(S^i, r) \leq \mathrm{w}(\mathcal{D}'_1, r) + (i+1) \cdot \mathrm{w}(\mathcal{D}'_2, r)$. Then, $S(\mathcal{D}_1, \mathcal{D}_2)$ can be obtained from S^n by applying a sequence of h applications of the rule m with rank k, and a sequence of t applications of the rule m with rank, respectively, $k_1, \dots, k_t$, in order to get Δ from $\Delta', \overset{n}{\dots}, \Delta'$. This means that $S(\mathcal{D}_1, \mathcal{D}_2) \lhd \Gamma, \Delta \vdash M'\{P/x_1, \dots, P/x_n\} : \tau$ and, by definition of forgetful substitution:

$$\begin{aligned}
M'\{P/x_1, \dots, P/x_n\} &= (M'[z/x_1, \dots, z/x_n])\{P/z\}\\
&= ((M'[z/x_1, \dots, z/x_n])[\mathtt{d}(x)/z])\{!P/x\} \qquad\qquad x \notin FV(M')\\
&= (M'[\mathtt{d}(x)/x_1, \dots, \mathtt{d}(x)/x_n])\{!P/x\} = M\{N/x\}.
\end{aligned}$$

By using the induction hypothesis, we finally have:

$$\begin{aligned}
\mathrm{w}(S(\mathcal{D}_1, \mathcal{D}_2), r) &= \mathrm{w}(S^n, r) + k \cdot h + \sum_{i=1}^{t} k_i\\
&\leq \mathrm{w}(\mathcal{D}'_1, r) + k \cdot \mathrm{w}(\mathcal{D}'_2, r) + k \cdot h + \sum_{i=1}^{t} k_i\\
&\leq \mathrm{w}(\mathcal{D}'_1, r) + r \cdot \mathrm{w}(\mathcal{D}'_2, r) + r \cdot h + \sum_{i=1}^{t} k_i\\
&\leq \mathrm{w}(\mathcal{D}_1, r) + (r \cdot (\mathrm{w}(\mathcal{D}'_2, r) + h) + 1 + \sum_{i=1}^{t} k_i)
\end{aligned}$$

$$
\begin{aligned}
&= \mathrm{w}(\mathcal{D}_1, r) + (\mathrm{w}(\mathcal{D}_2^*, r) + \sum_{i=1}^{t} k_i) \\
&\leq \mathrm{w}(\mathcal{D}_1, r) + \mathrm{w}(\mathcal{D}_2, r).
\end{aligned}
$$

This concludes the proof. □

We are now able to state the weighted version of the subject reduction property:

Theorem 72 (Weighted subject reduction). *Let $\mathcal{D} \triangleleft \Gamma \vdash M : \sigma$ and let $r \geq \mathrm{rk}(\mathcal{D})$. If $M \to M_1, M_2$ then there exist $\mathcal{D}_1$ and $\mathcal{D}_2$ such that:*

- $\mathcal{D}_i \triangleleft \Gamma \vdash M_i : \sigma$,
- $\mathrm{w}(\mathcal{D}_i, r) < \mathrm{w}(\mathcal{D}, r)$, *for* $i \in \{1, 2\}$.

Proof. The proof is by induction on the definition of the one-step reduction relation. We have several cases, and we consider the most interesting ones:

- If $M = (\lambda !x.N)!P \to N\{!P/x\} = M_1 = M_2$ then, by applying Lemma 68.(2) and Lemma 68.(3), $\mathcal{D}$ contains a derivation $\mathcal{D}^*$ of the form:

$$
\dfrac{\dfrac{\begin{array}{c}\mathcal{D}' \\ \Gamma', x : \tau \vdash N' : A\end{array}}{\Gamma' \vdash \lambda !x.N' : \tau \multimap A}\multimap\mathrm{I} \qquad \begin{array}{c}\mathcal{D}'' \\ \Gamma'' \vdash !P' : \tau\end{array}}{\Gamma', \Gamma'' \vdash (\lambda !x.N')!P' : A}\multimap\mathrm{E}
$$

  possibly followed by a sequence of applications of the rules ∀I, ∀E, and m. Let $t \geq 0$ be the number of applications of the rule m, and let $k_1, \dots, k_t$ be their respective rank. By applying Lemma 71, there exists a derivation $S(\mathcal{D}', \mathcal{D}'')$ such that $S(\mathcal{D}', \mathcal{D}'') \triangleleft \Gamma', \Gamma'' \vdash N'\{!P'/x\} : A$. We define $\mathcal{D}_1 = \mathcal{D}_2$ as the derivation obtained by applying to $S(\mathcal{D}', \mathcal{D}'')$ a sequence of applications of the rules ∀I, ∀E, and m in order to obtain $\Gamma \vdash N\{!P/x\} : \sigma$ as a concluding judgement. By Lemma 71, we have:

$$
\begin{aligned}
\mathrm{w}(\mathcal{D}_1, r) &= \mathrm{w}(S(\mathcal{D}', \mathcal{D}''), r) + \sum_{j=1}^{t} k_j \leq \mathrm{w}(\mathcal{D}', r) + \mathrm{w}(\mathcal{D}'', r) + \sum_{j=1}^{t} k_j \\
&< \mathrm{w}(\mathcal{D}', r) + \mathrm{w}(\mathcal{D}'', r) + \sum_{j=1}^{t} k_j + 2 = \mathrm{w}(\mathcal{D}, r).
\end{aligned}
$$

- If $M = \mathtt{proj}_B^{B \wedge B}(W_1, W_2) \to W_1 = M_1$ and $M = \mathtt{proj}_B^{B \wedge B}(W_1, W_2) \to W_2 = M_2$. By applying Lemma 44.(5)-(6), $\sigma = \forall \vec{\alpha}.(B'\langle D_1/\beta_1, \dots, D_n/\beta_n\rangle)$, where $\vec{\alpha} = \alpha_1, \dots, \alpha_k$, for some $k \geq 0$. Moreover, $\mathcal{D}$ is a derivation $\mathcal{D}^*$ of the form:

$$
\dfrac{\dfrac{\begin{array}{c}\mathcal{D}' \\ \vdash W_1 : B\end{array} \qquad \begin{array}{c}\mathcal{D}'' \\ \vdash W_2 : B\end{array}}{\vdash (W_1, W_2) : B \wedge B}\wedge\mathrm{I0}}{\vdash \mathtt{proj}_B^{B \wedge B}(W_1, W_2) : B}\wedge\mathrm{E}
$$

  followed by a sequence of applications of the rules ∀I, ∀E, and m. Then, we define $\mathcal{D}_1$ (resp. $\mathcal{D}_2$) as the derivation $\mathcal{D}'$ (resp. $\mathcal{D}''$) followed by the same sequence of rules ∀I, ∀E, and m, the latter being of rank 0 and introducing the context Γ. By definition of weight, we have: $\mathrm{w}(\mathcal{D}_1, r) = \mathrm{w}(\mathcal{D}', r) < \mathrm{w}(\mathcal{D}, r)$, and similarly for $\mathcal{D}_2$.

- If $M = \texttt{copy}_A^{W'}\ W\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (Q_1, Q_2) \to (Q_1[W/x_1], Q_2[W/x_2]) = M_1 = M_2$ then, by Lemma 68.(4), $\sigma = B_1 \wedge B_2$ and $\mathcal{D}$ is a derivation $\mathcal{D}^*$ of the form:

$$\dfrac{\begin{matrix}\mathcal{D}' \\ \Gamma' \vdash W : A\end{matrix} \quad \begin{matrix}\mathcal{D}'' \\ x_1 : A \vdash Q_1 : B_1\end{matrix} \quad \begin{matrix}\mathcal{D}''' \\ x_2 : A \vdash Q_2 : B_2\end{matrix} \quad \begin{matrix}\mathcal{D}'''' \\ \vdash W' : A\end{matrix}}{\Gamma' \vdash \texttt{copy}_A^{W'}\ W\ \texttt{as}\ x_1, x_2\ \texttt{in}\ (Q_1, Q_2) : B_1 \wedge B_2}\ \wedge\text{I1}$$

followed by a sequence of applications of the rule m. By Fact 55, Γ' is !-free, and hence all types in Γ' are linear. Then, since W is closed, Lemma 69 implies $\Gamma' = \emptyset$. Therefore, the applications of the rule m below $\mathcal{D}^*$ are all of rank 0, so that $\mathrm{w}(\mathcal{D}, r) = \mathrm{w}(\mathcal{D}^*, r)$. By applying Lemma 70 twice, there exist two derivations $S(\mathcal{D}', \mathcal{D}'') \triangleleft \vdash Q_1[W/x_1] : B_1$ and $S(\mathcal{D}', \mathcal{D}''') \triangleleft \vdash Q_2[W/x_2] : B_2$ such that $\mathrm{w}(S(\mathcal{D}', \mathcal{D}''), r) \leq \mathrm{w}(\mathcal{D}', r) + \mathrm{w}(\mathcal{D}'', r)$ and $\mathrm{w}(S(\mathcal{D}', \mathcal{D}'''), r) \leq \mathrm{w}(\mathcal{D}', r) + \mathrm{w}(\mathcal{D}''', r)$. We define $\mathcal{D}_1 = \mathcal{D}_2$ as the following derivation:

$$\cfrac{\dfrac{\begin{matrix}S(\mathcal{D}', \mathcal{D}'') \\ \vdash Q_1[W/x_1] : B_1\end{matrix} \qquad \begin{matrix}S(\mathcal{D}', \mathcal{D}''') \\ \vdash Q_2[W/x_2] : B_2\end{matrix}}{\vdash (Q_1[W/x_1], Q_2[W/x_2]) : B_1 \wedge B_2}\ \wedge\text{I0}}{\Gamma \vdash (Q_1[W/x_1], Q_2[W/x_2]) : B_1 \wedge B_2}\ m$$

By Proposition 56, we can safely assume that W' has largest size among the extended values with type A. Moreover, $\mathcal{D}'$ and $\mathcal{D}''''$ have no application of the rules sp and m so that, by Lemma 65.(3), $\mathrm{w}(\mathcal{D}', r) = |W| \leq |W'| = \mathrm{w}(\mathcal{D}'''', r)$. Therefore:

$$\begin{aligned}
\mathrm{w}(\mathcal{D}_1, r) &= \mathrm{w}(S(\mathcal{D}', \mathcal{D}''), r) + \mathrm{w}(S(\mathcal{D}', \mathcal{D}'''), r) + 1 \\
&\leq 2 \cdot \mathrm{w}(\mathcal{D}', r) + \mathrm{w}(\mathcal{D}'', r) + \mathrm{w}(\mathcal{D}''') + 1 \\
&\leq \mathrm{w}(\mathcal{D}', r) + \mathrm{w}(\mathcal{D}'', r) + \mathrm{w}(\mathcal{D}''', r) + \mathrm{w}(\mathcal{D}'''') + 1 \\
&< \mathrm{w}(\mathcal{D}', r) + \mathrm{w}(\mathcal{D}'', r) + \mathrm{w}(\mathcal{D}''', r) + \mathrm{w}(\mathcal{D}'''') + 2 = \mathrm{w}(\mathcal{D}^*, r) = \mathrm{w}(\mathcal{D}, r).
\end{aligned}$$

This concludes the proof. □

Corollary 73 (Unique surface distribution in $\mathsf{STA}_\oplus$). *Let $\mathcal{D} \triangleleft \Gamma \vdash M : \sigma$ in $\mathsf{STA}_\oplus$. Then there exists a unique surface distribution $\mathscr{D}$ such that $M \Rightarrow \mathscr{D}$.*

Proof. Let $r \geq \mathrm{rk}(\mathcal{D})$. We prove by induction on $\mathrm{w}(\mathcal{D}, r)$ that a derivation $\pi : M \Rightarrow \mathscr{D}$ exists, for some π and $\mathscr{D}$. If $M \in \mathrm{SNF}$ then, by applying rule $s1$, we have $M \Rightarrow M$. Otherwise, $M \to M_1, M_2$, for some M_1 and M_2. By Theorem 72 there exist $\mathcal{D}_1$, $\mathcal{D}_2$ such that $\mathcal{D}_i \triangleleft \Gamma \vdash M_i : \sigma$ and $\mathrm{w}(\mathcal{D}_i, r) < \mathrm{w}(\mathcal{D}, r)$, for all $i \in \{1, 2\}$. By applying the induction hypothesis, we have $\pi_1 : M_1 \Rightarrow \mathscr{D}_1$ and $\pi_2 : M_2 \Rightarrow \mathscr{D}_2$, for some π_1, π_2, $\mathscr{D}_1$, and $\mathscr{D}_2$. Finally, by applying rule $s2$:

$$\dfrac{M \to M_1, M_2 \qquad \pi_1 : M_1 \Rightarrow \mathscr{D}_1 \qquad \pi_2 : M_2 \Rightarrow \mathscr{D}_2}{M \Rightarrow \frac{1}{2} \cdot \mathscr{D}_1 + \frac{1}{2} \cdot \mathscr{D}_2}\ s2$$

Hence, for all $M \in \Lambda^!_{l,\oplus}$ typable in $\mathsf{STA}_\oplus$ there exists a surface distribution $\mathscr{D}$ such that $M \Rightarrow \mathscr{D}$, which is unique by Theorem 64. □

4.3.3 The Polytime Soundness Theorem

In defining the surface reduction for $\Lambda^!_{l,\oplus}$, we first introduced the one-step relation $\to$ between terms and pairs of terms, and then we lifted it to a relation $\Rightarrow$ between terms and distributions, essentially by turning $\mathtt{proj}_A^{A\wedge A}(W_1, W_2) \to W_1, W_2$ into $\mathtt{proj}_A^{A\wedge A}(W_1, W_2) \Rightarrow \frac{1}{2}\cdot W_1 + \frac{1}{2}\cdot W_2$.

Beside non-determinism, another important aspect of $\to$ is that it allows for different surface reduction strategies, as the following example shows.

Example 21. Consider the term $M \triangleq (\lambda !x.\langle \mathtt{ran}, \mathtt{d}(x)\rangle)!\mathbf{I}$, where $\mathtt{ran}$ is as in (4.17). If we first apply surface reduction to the innermost redex $\mathtt{ran}$ we obtain the pair of terms $(\lambda !x.\langle \mathtt{tt}, \mathtt{d}(x)\rangle)!\mathbf{I}$ and $(\lambda !x.\langle \mathtt{ff}, \mathtt{d}(x)\rangle)!\mathbf{I}$, that reduce in a single step to $\langle \mathtt{tt}, \mathbf{I}\rangle$ and $\langle \mathtt{ff}, \mathbf{I}\rangle$, respectively. If on the other hand we first apply the surface reduction to the outermost redex we obtain $\langle \mathtt{ran}, \mathbf{I}\rangle$, that reduces in one step either to $\langle \mathtt{tt}, \mathbf{I}\rangle$ or to $\langle \mathtt{ff}, \mathbf{I}\rangle$. Both surface reduction strategies are diagrammatically represented in Figure 4.13(a). For each such strategy is associated a derivation of $M \Rightarrow \frac{1}{2}\cdot\langle \mathtt{tt}, \mathbf{I}\rangle + \frac{1}{2}\cdot\langle \mathtt{ff}, \mathbf{I}\rangle$, as shown in Figure 4.13(b).

The example above shows that different surface reduction strategies can be applied to a term $M \in \Lambda^!_{l,\oplus}$. Moreover, if $M \Rightarrow \mathscr{D}$ holds for some surface distribution $\mathscr{D}$ (unique by Theorem 64), each surface reduction strategy corresponds to a specific derivation of $M \Rightarrow \mathscr{D}$. We are going to prove that, at least when M is typable in $\mathsf{STA}_\oplus$, all such derivations have the same size:

Lemma 74. *Let $\Gamma \vdash M : \sigma$ in $\mathsf{STA}_\oplus$. If $\pi' : M \Rightarrow \mathscr{D}$ and $\pi'' : M \Rightarrow \mathscr{D}$, then $|\pi'| = |\pi''|$.*

Proof. The proof is by induction on $|\pi'| + |\pi''|$. If the last rule of π' is $s1$ then M is a surface normal form, and the last rule of π'' must be $s1$. In this case, $|\pi'| = 0 = |\pi''|$. If the last rule of π' is $s2$, then M is not a surface normal form, so that the last rule of π'' is $s2$. Hence, π' and π'' have the following forms:

$$\dfrac{M \to M_1', M_2' \qquad \pi_1' : M_1' \Rightarrow \mathscr{D}_1' \qquad \pi_2' : M_2' \Rightarrow \mathscr{D}_2'}{\pi' : M \Rightarrow \mathscr{D}}\ s2$$

$$\dfrac{M \to M_1'', M_2'' \qquad \pi_1'' : M_1'' \Rightarrow \mathscr{D}_1'' \qquad \pi_2'' : M_2'' \Rightarrow \mathscr{D}_2''}{\pi'' : M \Rightarrow \mathscr{D}}\ s2$$

We have several possibilities depending on M_1', M_2', M_1'', M_2''. We just consider the case where they are all distinct. By applying Lemma 60 there exist N_1, N_2, N_3, N_4 such that $M_1' \to N_1, N_2$, $M_2' \to N_3, N_4$ and $\exists i \in \{1,2\}$ such that $M_i'' \to N_1, N_3$ and $M_{3-i}'' \to N_2, N_4$. Let us suppose $i = 1$. By Theorem 72 N_1, N_2, N_3 and N_4 are all typable in $\mathsf{STA}_\oplus$, and by Corollary 73, for all $1 \le j \le 4$, we have $\rho_j : N_j \Rightarrow \mathscr{E}_j$, for some ρ_j and $\mathscr{E}_j$. Then, we can construct the following derivations:

$$\dfrac{M_1' \to N_1, N_2 \qquad \rho_1 : N_1 \Rightarrow \mathscr{E}_1 \qquad \rho_2 : N_2 \Rightarrow \mathscr{E}_2}{\rho_1' : M_1' \Rightarrow \mathscr{D}_1'}\ s2$$

$$\dfrac{M_2' \to N_3, N_4 \qquad \rho_3 : N_3 \Rightarrow \mathscr{E}_3 \qquad \rho_4 : N_4 \Rightarrow \mathscr{E}_4}{\rho_2' : M_2' \Rightarrow \mathscr{D}_2'}\ s2$$

$$\dfrac{M_1'' \to N_1, N_3 \qquad \rho_1 : N_1 \Rightarrow \mathscr{E}_1 \qquad \rho_3 : N_3 \Rightarrow \mathscr{E}_3}{\rho_1'' : M_1'' \Rightarrow \mathscr{D}_1''}\ s2$$

$$\dfrac{M_2'' \to N_2, N_4 \qquad \rho_2 : N_2 \Rightarrow \mathscr{E}_2 \qquad \rho_4 : N_4 \Rightarrow \mathscr{E}_4}{\rho_2'' : M_2'' \Rightarrow \mathscr{D}_2''}\ s2$$

$$\begin{tikzcd}
 & (\lambda !x.\langle \mathtt{tt}, \mathtt{d}(x)\rangle)!\mathbf{I} \arrow[drr] & (\lambda !x.\langle \mathtt{ff}, \mathtt{d}(x)\rangle)!\mathbf{I} \arrow[r] & \langle \mathtt{ff}, \mathbf{I}\rangle \\
(\lambda !x.\langle \mathtt{ran}, \mathtt{d}(x)\rangle)!\mathbf{I} \arrow[ur] \arrow[urr] \arrow[r, dashed] & \langle \mathtt{ran}, \mathbf{I}\rangle \arrow[urr, dashed] \arrow[rr, dashed] & & \langle \mathtt{tt}, \mathbf{I}\rangle
\end{tikzcd}$$

(a) Different surface reduction strategies for $(\lambda !x.\langle \mathtt{ran}, \mathtt{d}(x)\rangle)!\mathbf{I}$.

$$\dfrac{(\lambda !x.\langle \mathtt{ran}, \mathtt{d}(x)\rangle)!\mathbf{I} \to (\lambda !x.\langle \mathtt{tt}, \mathtt{d}(x)\rangle)!\mathbf{I},\ (\lambda !x.\langle \mathtt{ff}, \mathtt{d}(x)\rangle)!\mathbf{I} \qquad \dfrac{(\lambda !x.\langle \mathtt{tt}, \mathtt{d}(x)\rangle)!\mathbf{I} \to \langle \mathtt{tt}, \mathbf{I}\rangle \qquad \overline{\langle \mathtt{tt}, \mathbf{I}\rangle \Rightarrow \langle \mathtt{tt}, \mathbf{I}\rangle}}{(\lambda !x.\langle \mathtt{tt}, \mathtt{d}(x)\rangle)!\mathbf{I} \Rightarrow \langle \mathtt{tt}, \mathbf{I}\rangle} \qquad \begin{array}{c}\vdots\\ (\lambda !x.\langle \mathtt{ff}, \mathtt{d}(x)\rangle)!\mathbf{I} \Rightarrow \langle \mathtt{ff}, \mathbf{I}\rangle\end{array}}{(\lambda !x.\langle \mathtt{ran}, \mathtt{d}(x)\rangle)!\mathbf{I} \Rightarrow \frac{1}{2}\cdot\langle \mathtt{tt}, \mathbf{I}\rangle + \frac{1}{2}\cdot\langle \mathtt{ff}, \mathbf{I}\rangle}$$

$$\dfrac{(\lambda !x.\langle \mathtt{ran}, \mathtt{d}(x)\rangle)!\mathbf{I} \to \langle \mathtt{ran}, \mathbf{I}\rangle \qquad \dfrac{\langle \mathtt{ran}, \mathbf{I}\rangle \to \langle \mathtt{tt}, \mathbf{I}\rangle, \langle \mathtt{ff}, \mathbf{I}\rangle \qquad \overline{\langle \mathtt{tt}, \mathbf{I}\rangle \Rightarrow \langle \mathtt{tt}, \mathbf{I}\rangle} \qquad \overline{\langle \mathtt{ff}, \mathbf{I}\rangle \Rightarrow \langle \mathtt{ff}, \mathbf{I}\rangle}}{\langle \mathtt{ran}, \mathbf{I}\rangle \Rightarrow \frac{1}{2}\cdot\langle \mathtt{tt}, \mathbf{I}\rangle + \frac{1}{2}\cdot\langle \mathtt{ff}, \mathbf{I}\rangle}}{(\lambda !x.\langle \mathtt{ran}, \mathtt{d}(x)\rangle)!\mathbf{I} \Rightarrow \frac{1}{2}\cdot\langle \mathtt{tt}, \mathbf{I}\rangle + \frac{1}{2}\cdot\langle \mathtt{ff}, \mathbf{I}\rangle}$$

(b) Different derivations of $(\lambda !x.\langle \mathtt{ran}, \mathtt{d}(x)\rangle)!\mathbf{I} \Rightarrow \frac{1}{2}\cdot\langle \mathtt{tt}, \mathbf{I}\rangle + \frac{1}{2}\cdot\langle \mathtt{ff}, \mathbf{I}\rangle$.

Figure 4.13: The term $(\lambda !x.\langle \mathtt{ran}, \mathtt{d}(x)\rangle)!\mathbf{I}$.

By applying the induction hypothesis we have:

$$\begin{aligned}
|\pi'| &= \max(|\pi_1'|, |\pi_2'|) + 1 \\
&= \max(|\rho_1'|, |\rho_2'|) + 1 \\
&= \max(\max(|\rho_1|, |\rho_2|) + 1, \max(|\rho_3|, |\rho_4|) + 1) + 1 \\
&= \max(\max(|\rho_1|, |\rho_3|) + 1, \max(|\rho_2|, |\rho_4|) + 1) + 1 \\
&= \max(|\rho_1''|, |\rho_2''|) + 1 \\
&= \max(|\pi_1''|, |\pi_2''|) + 1 = |\pi''|.
\end{aligned}$$

The remaining cases are similar. □

Remark 12. Consider two surface reduction strategies R and R' applied to some $M \in \Lambda^!_{l,\oplus}$ typable in $\mathsf{STA}_\oplus$ and such that all non-deterministic branches in R and R' reach a surface normal form. Lemma 74 says that, whenever n (resp. m) is the supremum of the set of the lengths of all non-deterministic branches of R (resp. R'), it must be $n = m$. This property depends on the fact that reduction is performed at a "surface level", namely out of the scope of any !, so that the reducts (i.e. the expressions to which redexes reduces) are never duplicable or erasable. A similar uniformity property holds in Simpson's linear λ-calculus, stating that all surface reductions strategies reaching a surface normal form starting from a given term M have the same length (see Section 2.2.3).

The lemma above allows us to speak about the "size" of $M \Rightarrow \mathscr{D}$. By Remark 12, the size of $M \Rightarrow \mathscr{D}$ gives an upper bound on the length of each non-deterministic branching of all possible reduction strategies applied to M. We now prove that such a bound can be taken as a polynomial in the size of M, from which we shall infer the Polytime Soundness Theorem (Theorem 76).

Lemma 75 (Strong polystep soundness). *Let $\mathcal{D} \triangleleft \Gamma \vdash_{\mathsf{STA}_\oplus} M : \sigma$ and let $\pi : M \Rightarrow \mathscr{D}$. Then:*

(1) $|\pi| \leq |M|^{\mathrm{d}(\mathcal{D})+1}$.

(2) For every reduction step $N \to N', N''$ that is premise of a rule $s2$ in π, $|N| \leq |M|^{\mathrm{d}(\mathcal{D})+1}$.

Proof. Let $\mathcal{D} \triangleleft \Gamma \vdash M : \sigma$. First, we observe that:

$$\begin{aligned}
\mathrm{w}(\mathcal{D}, \mathrm{rk}(\mathcal{D})) &\leq \mathrm{w}(\mathcal{D}, |M|) && \text{Lem. 65.(1)} \\
&\leq |M|^{\mathrm{d}(\mathcal{D})} \cdot \mathrm{w}(\mathcal{D}, 1) && \text{Lem. 65.(2)} \\
&= |M|^{\mathrm{d}(\mathcal{D})} \cdot |M| = |M|^{\mathrm{d}(\mathcal{D})+1}. && \text{Lem. 65.(3)}
\end{aligned}$$

Thus, to show both points, it suffices to prove by induction on the size of $\pi : M \Rightarrow \mathscr{D}$ that, for all $r \geq \mathrm{rk}(\mathcal{D})$:

(i) $|\pi| \leq \mathrm{w}(\mathcal{D}, r)$;

(ii) for every reduction step $N \to N', N''$ that is premise of a rule $s2$ in π, $|N| \leq \mathrm{w}(\mathcal{D}, r)$.

If the last rule of π is $s1$ then both points hold trivially. Otherwise, it ends with $s2$:

$$\dfrac{M \to M_1, M_2 \qquad \pi_1 : M_1 \Rightarrow \mathscr{D}_1 \qquad \pi_2 : M_2 \Rightarrow \mathscr{D}_2}{M \Rightarrow \frac{1}{2} \cdot \mathscr{D}_1 + \frac{1}{2} \cdot \mathscr{D}_2}\ s2$$

By Theorem 72, there exist $\mathcal{D}_1$ and $\mathcal{D}_2$ such that $\mathcal{D}_i \triangleleft \Gamma \vdash M_i : \sigma$ and $\mathrm{w}(\mathcal{D}_i, r) < \mathrm{w}(\mathcal{D}, r)$. As for point (i), by induction hypothesis, $|\pi_i| \leq \mathrm{w}(\mathcal{D}_i, r)$, with $i \in \{1, 2\}$. Hence, we have:

$$\begin{aligned} |\pi| &= \max(|\pi_1|, |\pi_2|) + 1 \\ &\leq \max(\mathrm{w}(\mathcal{D}_1, r), \mathrm{w}(\mathcal{D}_2, r)) + 1 \\ &\leq \mathrm{w}(\mathcal{D}, r). \end{aligned}$$

Concerning point (ii), by applying the induction hypothesis, for all $i \in \{1, 2\}$ and for all $N \to N_i', N_i''$ premise of a $s2$ in π_i, $|N| \leq \mathrm{w}(\mathcal{D}_i, r) < \mathrm{w}(\mathcal{D}, r)$. Moreover, by Lemma 65.(3), we have $|M| = \mathrm{w}(\mathcal{D}, 1) \leq \mathrm{w}(\mathcal{D}, r)$. □

Turing Machines are defined in Section 2.3.1. We now briefly recall their randomized formulations:

Definition 52 (Probabilistic Turing Machines)**.** A *Probabilistic Turing Machine* , PTM for short, is a Turing Machine with two transition functions δ_0 and δ_1, i.e. it is a tuple $\mathcal{P} \triangleq (\Gamma, Q, \delta_0, \delta_1)$ such that both (Γ, Q, δ_0) and (Γ, Q, δ_1) are Turing Machines. At each step in the computation the PTM chooses randomly which one of the transition functions δ_0 and δ_1 to apply (with equal probability $\frac{1}{2}$).

Definition 53. Let $\mathcal{P}$ be a PTM, and let $T : \mathbb{N} \longrightarrow \mathbb{N}$ be a function:

- we say that $\mathcal{P}$ *runs in $T(n)$-time* if its computation on every input x requires at most $T(|x|)$ steps, regardless of its random choices;
- we say that $\mathcal{P}$ *runs in $T(n)$-space* if its computation on every input x requires at most $T(|x|)$ cells of the tape, regardless of its random choices;

Remark 13. As pointed in [88], a β-reduction step $M \to_\beta M'$ can be simulated by a Turing Machine running in $\mathcal{O}(|M|^2)$-time. In a similar way, given a one-step reduction $M \to M_1, M_2$ in (4.15), we can build a PTM running in $\mathcal{O}(|M|^2)$-time that, when receiving in input (an encoding of) M, produces in output (an encoding of) M_i with probability a half.

We can now prove that $\mathsf{STA}_\oplus$ is sound with respect to the polynomial time PTMs.

Theorem 76 (Polytime soundness)**.** *Let $\mathcal{D} \triangleleft \Gamma \vdash M : \sigma$ be such that $\pi : M \Rightarrow \mathscr{D}$. Then there exists a PTM that runs in $\mathcal{O}(|M|^{3(\mathrm{d}(\mathcal{D})+1)})$-time such that, for all $S \in supp(\mathscr{D})$ with $\mathscr{D}(S) = p$, when receiving in input (an encoding of) M, it produces in output (an encoding of) the surface normal form S with probability p.*

Proof. By Lemma 75.(2) and by Remark 13, each reduction step $P \to P_1, P_2$ that is premise of a rule $s2$ in π can be simulated by a PTM running in $\mathcal{O}(|M|^{2(\mathrm{d}(\mathcal{D})+1)})$-time. By Lemma 75.(1) there can be at most $\mathcal{O}(|M|^{\mathrm{d}(\mathcal{D})+1})$ applications of $s2$ in π. By putting everything together, we obtain a PTM simulating the evaluation of M that runs in $\mathcal{O}(|M|^{3(\mathrm{d}(\mathcal{D})+1)})$-time. □

4.4 Polytime completeness

In this section we prove the Polytime Completeness Theorem for $\mathsf{STA}_\oplus$ (Theorem 91). The basic scheme of the proof is taken from Gaboardi and Ronchi Della Rocca [40], and consists in encoding PTMs configurations, transitions between configurations, the initialization of a PTM, and its output extraction. By putting everything together, we are able to represent in $\mathsf{STA}_\oplus$ a PTM running in polynomial time, where its transition function will be expressed by means of the non-deterministic rule $\mathtt{proj}_A^{A\wedge A}(W_1, W_2) \to W_1, W_2$ in (4.15). Before giving the complete encoding, we shall first show how to define in $\mathsf{STA}_\oplus$ boolean strings, natural numbers, iterations and polynomials.

Following Dal Lago and Toldin [28], we also prove that $\mathsf{STA}_\oplus$ is able to capture the complexity classes PP (Probabilistic Polynomial time) and BPP (Bounded-error Probabilistic Polynomial time). However, due to the presence of external error bounds, the characterization theorem for BPP will not be entirely in the style of ICC.

4.4.1 Strings, numerals and polynomial completeness

Gaboardi and Ronchi Della Rocca stressed in [40] that the presence of the multiplexor, i.e. rule m, makes the encoding of a Turing Machine "non-uniform" in STA. If we consider for example the standard type for natural numbers $\mathbf{N} \triangleq \forall\alpha.!(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha$, a term $\mathtt{succ}$ implementing the usual successor function with type $\mathbf{N} \multimap \mathbf{N}$ is unknown. This is why the usual data types are represented in STA by indexed families of types.

Definition 54 (Indexed numerals)**.** For all $i \geq 1$, the *indexed type* $\mathbf{N}_i$ and the *indexed numerals* $\underline{n}_i$ are defined as follows:

$$\mathbf{N}_i \triangleq \forall\alpha.!^i(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha$$
$$\underline{n}_i \triangleq \lambda!f.\lambda x.(\mathtt{d}^i(f).\overset{n}{\cdot\cdot}.(\mathtt{d}^i(f)x)) \qquad n \in \mathbb{N}$$

when $i = 1$, we shall write $\mathbf{N}$ (resp. $\underline{n}$) in place of $\mathbf{N}_i$ (resp. $\underline{n}_i$).

Proposition 77. *For all $i \geq 1$ and $n \in \mathbb{N}$, $\vdash_{\mathsf{STA}_\oplus} \underline{n}_i : \mathbf{N}_i$.*

Like the standard Church numerals, indexed numerals behave as iterators. In fact we can define n-long iterations of a term S (the *step* function) over a term B (the *base* function).

Definition 55. For all $i \geq 1$ and for all $A \in \Theta_{\wedge,!}$, the *indexed iterator* $\mathtt{iter}_i$ is defined as follows:

$$\mathtt{iter}_i \triangleq \lambda n.\lambda!s.\lambda x.n\,!^i(\mathtt{d}^i(s))\,x : \mathbf{N}_i \multimap !^i(A \multimap A) \multimap A \multimap A.$$

Proposition 78. *Let $i \geq 1$. If $\Delta \vdash B : A$ and $x_1 : A_1, \ldots, x_n : A_n \vdash S : A \multimap A$, then the following rule can be derived in $\mathsf{STA}_\oplus$:*

$$\frac{\Gamma \vdash \underline{n}_i : \mathbf{N}_i \qquad \Delta \vdash B : A \qquad x_1 : A_1, \ldots, x_n : A_n \vdash S : A \multimap A}{\Gamma, \Delta, y_1 : !^i A_1, \ldots, y_n : !^i A_n \vdash \mathtt{iter}_i\, \underline{n}_i\, (!^i S[\mathtt{d}^i(y_1)/x_1, \ldots, \mathtt{d}^i(y_n)/x_n])\, B : A}$$

Moreover, for every indexed numeral $\underline{n}_i$, we have:

$$\mathtt{iter}_i\, \underline{n}_i\, (!^i S^*)\, B \Rightarrow S^*(.\overset{n}{\cdot\cdot}.(S^* B)\ldots)$$

where $S^* \triangleq (S[\mathtt{d}^i(y_1)/x_1, \ldots, \mathtt{d}^i(y_n)/x_n])$.

Remark 14. The step function can be iterated only if it is definable through a term typable with type $A \multimap A$, for some linear type A. This is in contrast with what happens in linear logic, where, in general, step functions proving $!A \multimap A$ are allowed.

Definition 56. Let $i, j \geq 1$. The indexed successor, addition, and multiplication are definable in $\mathsf{STA}_\oplus$ as follows:

- $\mathtt{succ}_i \triangleq \lambda n.\lambda!f.\lambda x.\mathtt{d}^{i+1}(f)(n\,(!^i\mathtt{d}^{i+1}(f))\,x)$;

- $\texttt{add}_{i,j} \triangleq \lambda n.\lambda m.\lambda !f.\lambda x.n\,(!^i\texttt{d}^{\max(i,j)+1}(f))(m\,(!^j\texttt{d}^{(\max(i,j)+1}(f))\,x)$;
- $\texttt{mult}_{i,j} \triangleq \lambda n.\lambda m.\lambda !f.n\,!^i(m\,(!^j\texttt{d}^{i+j}(f)))$.

Proposition 79. *For all $i, j \geq 1$:*

- $\vdash_{\mathsf{STA}_\oplus} \texttt{succ}_i : \mathbf{N}_i \multimap \mathbf{N}_{i+1}$*;*
- $\vdash_{\mathsf{STA}_\oplus} \texttt{add}_{i,j} : \mathbf{N}_i \multimap \mathbf{N}_j \multimap \mathbf{N}_{\max(i,j)+1}$*;*
- $\vdash_{\mathsf{STA}_\oplus} \texttt{mult}_{i,j} : \mathbf{N}_i \multimap !^i\mathbf{N}_j \multimap \mathbf{N}_{i+j}$*.*

A straightforward consequence of Remark 14 and Proposition 79 is that the successor, the addition, and the multiplication of Definition 56 cannot be iterated, they can only be composed to obtain all polynomials.

Theorem 80 (Representing polynomial functions [40])**.** *Let $p : \mathbb{N} \longrightarrow \mathbb{N}$ be a polynomial in the variable* x *and* $\deg(p)$ *be its degree. There is $\underline{p}$ such that:*

$$x : !^{\deg(p)}\mathbf{N} \vdash_{\mathsf{STA}_\oplus} \underline{p} : \mathbf{N}_{2\deg(p)+1}$$

Proof. Consider p in Horner normal form, i.e. $p = a_0 + \mathrm{x} \cdot (a_1 + \mathrm{x} \cdot (\ldots(a_{n-1} + \mathrm{x} \cdot a_n)\ldots))$. By induction on $\deg(p)$ we show something stronger, namely that the following is derivable for all $i > 0$:

$$x_0 : \mathbf{N}_i, x_1 : !^i\mathbf{N}_i, \ldots, x_n : !^{i(\deg(p^*)-1)}\mathbf{N}_i \vdash \underline{p^*} : \mathbf{N}_{i\deg(p^*)+\deg(p^*)+1} \tag{4.18}$$

where $p^* \triangleq a_0 + \mathrm{x}_0 \cdot (a_1 + \mathrm{x}_1 \cdot (\ldots(a_{n-1} + \mathrm{x}_n \cdot a_n)\ldots))$. The base case is trivial, so consider $p^* = a_0 + \mathrm{x}_0 \cdot p'$. By induction hypothesis:

$$x_1 : \mathbf{N}_i, \ldots, x_n : !^{i(\deg(p')-1)}\mathbf{N}_i \vdash \underline{p'} : \mathbf{N}_{i\cdot\deg(p')+\deg(p')+1}.$$

If $k \triangleq i \cdot \deg(p') + \deg(p') + 1$ we define:

$$\underline{p^*} \triangleq \texttt{add}_{1,i+k}\,\underline{a_0}\,(\texttt{mult}_{i,k}\,x_0\,(!^i\underline{p'}[\texttt{d}^i(x_1)/x_1, \ldots, \texttt{d}^i(x_n)/x_n])).$$

We have:

$$x_0 : \mathbf{N}_i, x_1 : !^i\mathbf{N}_i, \ldots, x_n : !^{i(\deg(p')-1)+i}\mathbf{N}_i \vdash \underline{p^*} : \mathbf{N}_{i(\deg(p')+1)+\deg(p')+1+1}.$$

Since $\deg(p^*) = \deg(p') + 1$, the above judgement is exactly (4.18). Now, by taking $i = 1$ and repeatedly applying the rule m:

$$x : !^{\deg(p)}\mathbf{N} \vdash \underline{p} : \mathbf{N}_{2\deg(p)+1}.$$

where $\underline{p} \triangleq \underline{p^*}[\texttt{d}^{h_1}(x)/x_1 \ldots \texttt{d}^{h_n}(x)/x_n]$, for some $h_1, \ldots, h_n \geq 0$. □

Booleans are defined in (3.2). By convention, we shall fix:

$$\underline{0} \triangleq \texttt{tt} \qquad\qquad \underline{1} \triangleq \texttt{ff}. \tag{4.19}$$

In Figure 3.9 of the previous chapter we show how to encode the boolean functions of the standard unbounded fan-in basis in IMLL_2. As a consequence, every boolean function can be represented in this system, and hence in $\mathsf{STA}_\oplus$.

Lemma 81 (Functional completeness [65])**.** *Let $n, m \in \mathbb{N}$. Every boolean total function $f : \{0,1\}^n \longrightarrow \{0,1\}^m$ is represented by a term $\underline{f}$ such that $\vdash_{\mathsf{IMLL}_2} \underline{f} : \mathbf{B}^n \multimap \mathbf{B}^m$.*

As in the case of numerals, the encoding of strings of booleans requires the introduction of indexes:

Definition 57 (Indexed strings)**.** For all $i \geq 1$, the *indexed type* $\mathbf{S}_i$ and the *indexed n-ary boolean strings* $\underline{s}_i$ are defined as follows:

$$\mathbf{S}_i \triangleq \forall \alpha.!^i(\mathbf{B} \multimap \alpha \multimap \alpha) \multimap (\alpha \multimap \alpha)$$

$$\underline{s}_i \triangleq \lambda !c.\lambda z.\mathtt{d}^i(c)\underline{b_1}(\dots(\mathtt{d}^i(c)\underline{b_n}z)\dots) \qquad \text{where } s = b_1 \dots b_n \in \{0,1\}^n \text{ and } n \in \mathbb{N}.$$

when $n = 1$, we shall write $\mathbf{S}$ (resp. $\underline{s}$) in place of $\mathbf{S}_1$ (resp. $\underline{s}_1$).

Proposition 82. *For all $i \geq 1$, $\vdash_{\mathsf{STA}_\oplus} \underline{s}_i : \mathbf{S}_i$.*

The function associating with each string of booleans its length can be defined for all $i \geq 1$ as follows:

$$\mathtt{len}_i \triangleq \lambda s.\lambda f.s\,!^i(\lambda x.\lambda y.\mathtt{let}\ \mathtt{E_B}\,x\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ fy) \tag{4.20}$$

with type $\mathbf{S}_i \multimap \mathbf{N}_i$, where $\mathtt{E_B}$ is as in (3.3).

4.4.2 Encoding the polytime PTM

In this subsection we show how to encode the polytime PTM in $\mathsf{STA}_\oplus$ and how to simulate its computation by means of the relation $\Rightarrow$ in Definition 48. One of the key steps toward completeness is to prove that every PTM transition function is definable in $\mathsf{STA}_\oplus$. First, we show how to encode the transition function of a (deterministic) Turing Machine.

Proposition 83 (Transition functions)**.** *The transition function $\delta_\mathcal{M}$ of a Turing Machine $\mathcal{M}$ with at most 2^n states is represented in $\mathsf{STA}_\oplus$ by a suitable $\underline{\delta_\mathcal{M}} : \mathbf{B}^{n+1} \multimap \mathbf{B}^{n+2}$.*

Proof. Let $\mathcal{M}$ be a Turing Machine with alphabet $\Gamma = \{0,1\}$ and states $Q = \{q_1, \dots, q_k\}$, where $k \leq 2^n$. We use the inhabitants of $\mathbf{B}$ to represent both the elements of Γ and the head moves, while the inhabitants of $\mathbf{B}^n$ are used to represent states in Q. A configuration $(q, b) \in Q \times \Gamma$ is then encoded by the pair $\langle \underline{q}, \underline{b} \rangle$ of type $\mathbf{B}^{n+1}$. The triple $\langle \underline{q'}, \underline{b'}, \underline{m} \rangle$ of type $\mathbf{B}^{n+2}$ stands for $\delta_\mathcal{M}((b, q)) = (q', b', m) \in Q \times \Gamma \times \{\text{left}, \text{right}\}$. Now, we want the transition function $\delta_\mathcal{M}$ to be represented by a term $\underline{\delta_\mathcal{M}}$ of type $\mathbf{B}^{n+1} \multimap \mathbf{B}^{n+2}$. So, let:

$$M_{\langle \mathtt{t}^n, \mathtt{t} \rangle}, M_{\langle \mathtt{t}^{n-1}\mathtt{f}, \mathtt{t} \rangle}, \dots, M_{\langle \mathtt{f}^{n-1}\mathtt{t}, \mathtt{t} \rangle}, M_{\langle \mathtt{f}^n, \mathtt{t} \rangle}, M_{\langle \mathtt{t}^n, \mathtt{f} \rangle}, M_{\langle \mathtt{t}^{n-1}\mathtt{f}, \mathtt{f} \rangle}, \dots, M_{\langle \mathtt{f}^{n-1}\mathtt{t}, \mathtt{f} \rangle}, M_{\langle \mathtt{f}^n, \mathtt{f} \rangle}$$

be a family of (not necessarily distinct) inhabitants of $\mathbf{B}^{n+2}$ indexed by the inhabitants of $\mathbf{B}^{n+1}$, and such that:

$$M_{\langle \underline{q}, \underline{b} \rangle} = \langle \underline{q'}, \underline{b'}, \underline{m} \rangle \text{ if and only if } \delta_\mathcal{M}((q, b)) = (q', b', m).$$

By Definition 16 in Section 3.2.2 we can define:

$$\begin{aligned}\underline{\delta_\mathcal{M}} \triangleq \lambda x.\mathtt{if}\ x\ \mathtt{then}\ \big[&M_{\langle \mathtt{t}^n, \mathtt{t} \rangle}, M_{\langle \mathtt{t}^{n-1}\mathtt{f}, \mathtt{t} \rangle}, \dots, M_{\langle \mathtt{f}^{n-1}\mathtt{t}, \mathtt{t} \rangle}, M_{\langle \mathtt{f}^n, \mathtt{t} \rangle},\\ &M_{\langle \mathtt{t}^n, \mathtt{f} \rangle}, M_{\langle \mathtt{t}^{n-1}\mathtt{f}, \mathtt{f} \rangle}, \dots, M_{\langle \mathtt{f}^{n-1}\mathtt{t}, \mathtt{f} \rangle}, M_{\langle \mathtt{f}^n, \mathtt{f} \rangle}\big] : \mathbf{B}^{n+1} \multimap \mathbf{B}^{n+2}\end{aligned}$$

so that, $\underline{\delta_\mathcal{M}} \langle \underline{q}, \underline{b} \rangle \Rightarrow \langle \underline{q'}, \underline{b'}, \underline{m} \rangle$ if and only if $\langle \underline{q'}, \underline{b'}, \underline{m} \rangle = M_{\langle \underline{q}, \underline{b} \rangle}$ if and only if $\delta_\mathcal{M}((q, b)) = (q', b', m)$. □

A Probabilistic Turing Machine $\mathcal{P}$ is a Turing Machine whose transition function $\delta_{\mathcal{P}}$ can be seen as the superposition of two (deterministic) Turing Machine transition functions δ_0 and δ_1: at each step in the computation, $\mathcal{P}$ selects δ_0 with probability $\frac{1}{2}$ and δ_1 with probability $\frac{1}{2}$. So, let $\delta_0, \delta_1 : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\text{left}, \text{right}\}$ be two Turing Machine transition functions, where Q contains at most 2^n states and $\Gamma = \{0, 1\}$. By Proposition 83, there exist $\underline{\delta_0}$ and $\underline{\delta_1}$, both having type $\mathbf{B}^{n+1} \multimap \mathbf{B}^{n+2}$ in $\mathsf{STA}_{\oplus}$. Then, we define:

$$\underline{\delta_{\mathcal{P}}} \triangleq \lambda x.\mathtt{proj}^{\mathbf{B}^{n+2}\wedge\mathbf{B}^{n+2}}_{\mathbf{B}^{n+2}}(\mathtt{copy}^{\mathtt{tt}^{n+1}}_{\mathbf{B}^{n+1}}\, x \;\mathtt{as}\; x_0, x_1 \;\mathtt{in}\; (\underline{\delta_0}\, x_0, \underline{\delta_1}\, x_1))$$

whose typing is shown in Figure 4.14(a). Moreover, if $\langle \underline{q}, \underline{p} \rangle$ is a pair encoding a PTM configuration $(q, b) \in Q \times \Gamma$, and if $\underline{\delta_i}\langle \underline{q}, \underline{p} \rangle \Rightarrow \langle \underline{q_i}, \underline{b_i}, \underline{m_i} \rangle$ for $i \in \{1, 2\}$, then:

$$\underline{\delta_{\mathcal{P}}}\langle \underline{q}, \underline{p} \rangle \Rightarrow \frac{1}{2} \cdot \langle \underline{q_0}, \underline{b_0}, \underline{m_0} \rangle + \frac{1}{2} \cdot \langle \underline{q_1}, \underline{b_1}, \underline{m_1} \rangle$$

whose derivation is shown in Figure 4.14(b).

Summing up, we have:

Corollary 84 (Probabilistic transition functions)**.** *The transition function $\delta_{\mathcal{P}}$ of a Probabilistic Turing Machine $\mathcal{P}$ with at most 2^n states is definable in $\mathsf{STA}_{\oplus}$ by a suitable $\underline{\delta_{\mathcal{P}}} : \mathbf{B}^{n+1} \multimap \mathbf{B}^{n+2}$.*

A configuration can be represented by a tuple divided up in three parts: the first one represents the left hand-side of the tape with respect to the head; the second one represents the right part of the tape starting with the cell scanned by the head; finally, the third part represents the state of the machine. W.l.o.g., we shall assume that the left part of the tape is represented in reversed order, that the alphabet is composed by the two symbols 0 and 1, and that the final states are divided into accepting and rejecting.

Definition 58 (Indexed configuration)**.** For all $i, k \geq 1$, we define the *indexed type* $\mathbf{PTM}_i$ and the *indexed configuration* $\mathtt{config}_i$ as follows:

$$\mathbf{PTM}^k_i \triangleq \forall\alpha.!^i(\mathbf{B} \multimap \alpha \multimap \alpha) \multimap ((\alpha \multimap \alpha)^2 \otimes \mathbf{B}^k)$$
$$\mathtt{config}_i \triangleq \lambda!c.\langle \mathtt{d}^i(c)\, \underline{b^l_0} \circ \cdots \circ \mathtt{d}^i(c)\, \underline{b^l_n}\,,\; \mathtt{d}^i(c)\, \underline{b^r_0} \circ \cdots \circ \mathtt{d}^i(c)\, \underline{b^r_m}\,,\; \underline{Q} \rangle.$$

where $M \circ N \triangleq \lambda z.M(Nz)$, $\underline{Q} \triangleq \langle \underline{q_1}, \ldots, \underline{q_k} \rangle$, and $b^l_0, \ldots, b^l_n, b^r_0, \ldots, b^r_m, q_1, \ldots, q_k \in \{0, 1\}$, for $n, m \in \mathbb{N}$.

Proposition 85. *For all $i, k \geq 1$, $\vdash_{\mathsf{STA}_{\oplus}} \mathtt{config}_i : \mathbf{PTM}^k_i$.*

In the above definition, the terms:

$$\mathtt{d}^i(c)\, \underline{b^l_0} \circ \cdots \circ \mathtt{d}^i(c)\, \underline{b^l_{n_l}} \qquad \mathtt{d}^i(c)\, \underline{b^r_0} \circ \cdots \circ \mathtt{d}^i(c)\, \underline{b^r_{n_r}} \qquad \underline{Q} \triangleq \langle \underline{q_1}, \ldots, \underline{q_k} \rangle$$

represent, respectively, the left and the right part of the tape, where $\mathtt{d}^i(c)\, \underline{b^r_0}$ is the scanned symbol, and the current state $Q = (q_1, \ldots, q_k)$.

Following Mairson and Terui [65], in order to show that the PTM transition from a configuration to another is definable we consider two distinct phases. In the first one, the PTM configuration is decomposed to extract the first symbol of each part of the tape. In the second phase, depending on the transitions function, these symbols are combined to reconstruct the tape after the transition step. Thus, we require an intermediate type, denoted $\mathbf{ID}^k_i$, and defined for all $i, k \geq 1$ as follows:

$$\mathbf{ID}^k_i \triangleq \forall\alpha.!^i(\mathbf{B} \multimap \alpha \multimap \alpha) \multimap ((\alpha \multimap \alpha)^2 \otimes (\mathbf{B} \multimap \alpha \multimap \alpha) \otimes \mathbf{B} \otimes (\mathbf{B} \multimap \alpha \multimap \alpha) \otimes \mathbf{B} \otimes \mathbf{B}^k)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x:\mathbf{B}^{n+1}\vdash x:\mathbf{B}^{n+1}}\,ax \quad \begin{array}{c}\vdots\\ x_0:\mathbf{B}^{n+1}\vdash \underline{\delta_0}\,x_0:\mathbf{B}^{n+2}\end{array} \quad \begin{array}{c}\vdots\\ x_1:\mathbf{B}^{n+1}\vdash \underline{\delta_1}\,x_1:\mathbf{B}^{n+2}\end{array} \quad \begin{array}{c}\vdots\\ \vdash \mathtt{tt}^{n+1}:\mathbf{B}^{n+1}\end{array}}{x:\mathbf{B}^{n+1}\vdash \mathtt{copy}^{\mathtt{tt}^{n+1}}_{\mathbf{B}^{n+1}}\,x\ \mathtt{as}\ x_0,x_1\ \mathtt{in}\ (\underline{\delta_0}\,x_0,\underline{\delta_1}x_1):\mathbf{B}^{n+2}\wedge\mathbf{B}^{n+2}}\,\wedge\mathrm{I1}}{x:\mathbf{B}^{n+1}\vdash \mathtt{proj}^{\mathbf{B}^{n+2}\wedge\mathbf{B}^{n+2}}_{\mathbf{B}^{n+2}}(\mathtt{copy}^{\mathtt{tt}^{n+1}}_{\mathbf{B}^{n+1}}\,x\ \mathtt{as}\ x_0,x_1\ \mathtt{in}\ (\underline{\delta_0}\,x_0,\underline{\delta_1}\,x_1)):\mathbf{B}^{n+2}}\,\wedge\mathrm{E}}{\vdash \lambda x.\mathtt{proj}^{\mathbf{B}^{n+2}\wedge\mathbf{B}^{n+2}}_{\mathbf{B}^{n+2}}(\mathtt{copy}^{\mathtt{tt}^{n+1}}_{\mathbf{B}^{n+1}}\,x\ \mathtt{as}\ x_0,x_1\ \mathtt{in}\ (\underline{\delta_0}\,x_0,\underline{\delta_1}\,x_1)):\mathbf{B}^{n+1}\multimap\mathbf{B}^{n+2}}\,\multimap\mathrm{I}$$

(a) A derivation in $\mathsf{STA}_{\oplus}$ for $\underline{\delta_{\mathcal{P}}} \triangleq \lambda x.\mathtt{proj}^{\mathbf{B}^{n+2}\wedge\mathbf{B}^{n+2}}_{\mathbf{B}^{n+2}}(\mathtt{copy}^{\mathtt{tt}^{n+1}}_{\mathbf{B}^{n+1}}\,x\ \mathtt{as}\ x_0,x_1\ \mathtt{in}\ (\underline{\delta_0}\,x_0,\underline{\delta_1}\,x_1))$.

$$\dfrac{\underline{\delta_{\mathcal{P}}}\langle\underline{q},\underline{p}\rangle \to \delta_{\mathcal{P}}^{\langle\underline{q},\underline{p}\rangle} \qquad \dfrac{\delta_{\mathcal{P}}^{\langle\underline{q},\underline{p}\rangle}\to\delta_0^{\langle\underline{q},\underline{p}\rangle}\oplus\delta_1^{\langle\underline{q},\underline{p}\rangle} \qquad \dfrac{\delta_0^{\langle\underline{q},\underline{p}\rangle}\oplus\delta_1^{\langle\underline{q},\underline{p}\rangle}\to\underline{\delta_0}\langle\underline{q},\underline{p}\rangle,\underline{\delta_1}\langle\underline{q},\underline{p}\rangle \quad \begin{array}{c}\vdots\\ \underline{\delta_0}\langle\underline{q},\underline{p}\rangle\Rightarrow\langle\underline{q_0},\underline{b_0},\underline{m_0}\rangle\end{array} \quad \begin{array}{c}\vdots\\ \underline{\delta_1}\langle\underline{q},\underline{p}\rangle\Rightarrow\langle\underline{q_1},\underline{b_1},\underline{m_1}\rangle\end{array}}{\delta_0^{\langle\underline{q},\underline{p}\rangle}\oplus\delta_1^{\langle\underline{q},\underline{p}\rangle}\Rightarrow\frac{1}{2}\cdot\langle\underline{q_0},\underline{b_0},\underline{m_0}\rangle+\frac{1}{2}\cdot\langle\underline{q_1},\underline{b_1},\underline{m_1}\rangle}\,s2}{\delta_{\mathcal{P}}^{\langle\underline{q},\underline{p}\rangle}\Rightarrow\frac{1}{2}\cdot\langle\underline{q_0},\underline{b_0},\underline{m_0}\rangle+\frac{1}{2}\cdot\langle\underline{q_1},\underline{b_1},\underline{m_1}\rangle}\,s2}{\underline{\delta_{\mathcal{P}}}\langle\underline{q},\underline{p}\rangle\Rightarrow\frac{1}{2}\cdot\langle\underline{q_0},\underline{b_0},\underline{m_0}\rangle+\frac{1}{2}\cdot\langle\underline{q_1},\underline{b_1},\underline{m_1}\rangle}\,s2$$

(b) Derivation of $\underline{\delta_{\mathcal{P}}}\langle\underline{q},\underline{p}\rangle\Rightarrow\frac{1}{2}\cdot\langle\underline{q_0},\underline{b_0},\underline{m_0}\rangle+\frac{1}{2}\cdot\langle\underline{q_1},\underline{b_1},\underline{m_1}\rangle$.

Abbreviations:

$$\delta_{\mathcal{P}}^{\langle\underline{q},\underline{p}\rangle} \triangleq \mathtt{proj}^{\mathbf{B}^{n+2}\wedge\mathbf{B}^{n+2}}_{\mathbf{B}^{n+2}}(\mathtt{copy}^{\mathtt{tt}^{n+1}}_{\mathbf{B}^{n+1}}\,\langle\underline{q},\underline{p}\rangle\ \mathtt{as}\ x_0,x_1\ \mathtt{in}\ (\underline{\delta_0}\,x_0,\underline{\delta_1}\,x_1))$$

$$\delta_0^{\langle\underline{q},\underline{p}\rangle}\oplus\delta_1^{\langle\underline{q},\underline{p}\rangle} \triangleq \mathtt{proj}^{\mathbf{B}^{n+2}\wedge\mathbf{B}^{n+2}}_{\mathbf{B}^{n+2}}((\underline{\delta_0}\langle\underline{q},\underline{p}\rangle,\underline{\delta_1}\langle\underline{q},\underline{p}\rangle)).$$

Figure 4.14: The encoding of a PTM transition function.

and the decomposition phase is represented by the following term:

$$\begin{aligned}\mathtt{decom}_i \triangleq\ & \lambda m.\lambda !c.\mathtt{let}\ m\,!^i(F[\mathtt{d}^i(c)])\ \mathtt{be}\ l,r,q\ \mathtt{in}\\ & (\mathtt{let}\ l\langle \mathbf{I}, \lambda x.\mathtt{let}\ \mathtt{E}_{\mathbf{B}}\,x\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ \mathbf{I}, \underline{0}\rangle\ \mathtt{be}\ s_l, c_l, b_0^l\ \mathtt{in}\\ & (\mathtt{let}\ r\langle \mathbf{I}, \lambda x.\mathtt{let}\ \mathtt{E}_{\mathbf{B}}\,x\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ \mathbf{I}, \underline{0}\rangle\ \mathtt{be}\ s_r, c_r, b_0^r\ \mathtt{in}\\ & \langle s_l, s_r, c_l, b_0^l, c_r, b_0^r, q\rangle))\end{aligned} \tag{4.21}$$

where $F[x] \triangleq \lambda b.\lambda z.\mathtt{let}\ z\ \mathtt{be}\ g,h,i\ \mathtt{in}\ \langle hi \circ g, x, b\rangle$ and $\mathtt{E}_{\mathbf{B}}$ is as in (3.3).

Proposition 86. *For all $i, k \geq 1$, $\vdash_{\mathsf{STA}_\oplus} \mathtt{decom}_i : \mathbf{PTM}_i^k \multimap \mathbf{ID}_i^k$.*

The behaviour of $\mathtt{decom}_i$ is to decompose a configuration in such a way as to extract the symbols of the tape which determine, together with the current state, the structure of the next configuration:

$$\begin{aligned}&\mathtt{decom}_i(\lambda !c.\langle \mathtt{d}^i(c)\,\underline{b_0^l} \circ \cdots \circ \mathtt{d}^i(c)\,\underline{b_n^l}\,,\ \mathtt{d}^i(c)\,\underline{b_0^r} \circ \cdots \circ \mathtt{d}^i(c)\,\underline{b_m^r}\,,\ \underline{Q}\rangle)\\ &\quad \Rightarrow \lambda !c.\langle \mathtt{d}^i(c)\,\underline{b_1^l} \circ \cdots \circ \mathtt{d}^i(c)\,\underline{b_n^l}\,,\ \mathtt{d}^i(c)\,\underline{b_1^r} \circ \cdots \circ \mathtt{d}^i(c)\,\underline{b_m^r}\,,\ \mathtt{d}^i(c)\,,\ \underline{b_0^l}\,,\ \mathtt{d}^i(c)\,,\ \underline{b_0^r}\,,\ \underline{Q}\rangle.\end{aligned}$$

Analogously, the composition phase is represented by the following term:

$$\begin{aligned}\mathtt{com}_i \triangleq\ & \lambda s.\lambda !c.\mathtt{let}\ s\,!^i(\mathtt{d}^i(c))\ \mathtt{be}\ l,r,c_l,b_l,c_r,b_r,q\ \mathtt{in}\ \mathtt{let}\ \underline{\delta_{\mathcal{P}}}\,\langle b_r, q\rangle\ \mathtt{be}\ q', b', m\ \mathtt{in}\\ & (\mathtt{if}\ m\ \mathtt{then}\ M_1\ \mathtt{else}\ M_2) b' q' \langle l, r, c_l, b_l, c_r\rangle\end{aligned} \tag{4.22}$$

where $\delta_{\mathcal{P}}$ is the transition function of the PTM $\mathcal{P}$ as in Corollary 84, and:

$$\begin{aligned}M_1 &\triangleq \lambda b'.\lambda q'.\lambda p.\mathtt{let}\ p\ \mathtt{be}\ l,r,c_l,b_l,c_r\ \mathtt{in}\ \langle c_r\,b' \circ c_l\,b_l \circ l, r, q'\rangle\\ M_2 &\triangleq \lambda b'.\lambda q'.\lambda p.\mathtt{let}\ p\ \mathtt{be}\ l,r,c_l,b_l,c_r\ \mathtt{in}\ \langle l, c_l\,b_l \circ c_r\,b' \circ r, q'\rangle.\end{aligned}$$

Proposition 87. *For all $i, k \geq 1$, $\vdash_{\mathsf{STA}_\oplus} \mathtt{com}_i : \mathbf{ID}_i^k \multimap \mathbf{PTM}_i^k$.*

Then, the behaviour of $\mathtt{com}_i$, depending on the $\delta_{\mathcal{P}}$ transition function and on the current state, is to combine the symbols we put aside in order to return a distribution of the next configurations. For example, if the deterministic transition functions δ_0 and δ_1 defining $\delta_{\mathcal{P}}$ are such that $\delta_0((b_0^r, Q)) = (Q', b', \text{right}))$ and $\delta_1((b_0^r, Q)) = (Q'', b'', \text{left})$, then:

$$\begin{aligned}&\mathtt{com}_i\,(\langle \mathtt{d}^i(c)\,\underline{b_1^l} \circ \cdots \circ \mathtt{d}^i(c)\,\underline{b_n^l}\,,\ \mathtt{d}^i(c)\,\underline{b_1^r} \circ \cdots \circ \mathtt{d}^i(c)\,\underline{b_m^r}\,,\ \mathtt{d}^i(c)\,,\ \underline{b_0^l}\,,\ \mathtt{d}^i(c)\,,\ \underline{b_0^r}\,,\ \underline{Q}\rangle)\\ &\quad \Rightarrow \frac{1}{2}\cdot \lambda !c.\langle \mathtt{d}^i(c)\,\underline{b'} \circ \mathtt{d}^i(c)\,\underline{b_0^l} \circ \mathtt{d}^i(c)\,\underline{b_1^l} \circ \cdots \circ \mathtt{d}^i(c)\,\underline{b_n^l}, \mathtt{d}^i(c)\,\underline{b_1^r} \circ \cdots \circ \mathtt{d}^i(c)\,\underline{b_m^r}, \underline{Q'}\rangle\\ &\qquad\qquad + \\ &\quad \frac{1}{2}\cdot \lambda !c.\langle \mathtt{d}^i(c)\,\underline{b_1^l} \circ \cdots \circ \mathtt{d}^i(c)\,\underline{b_n^l}, \mathtt{d}^i(c)\,\underline{b_0^l} \circ \mathtt{d}^i(c)\,\underline{b''} \circ \mathtt{d}^i(c)\,\underline{b_1^r} \circ \cdots \circ \mathtt{d}^i(c)\,\underline{b_m^r}, \underline{Q'}\rangle.\end{aligned}$$

By combining the above terms we obtain an entire PTM transition step.

Definition 59 (Indexed transition step)**.** Let $i, k \geq 1$. The *indexed transition step* is defined by $\mathtt{tr}_i \triangleq \mathtt{com}_i \circ \mathtt{decom}_i$, with type $\mathbf{PTM}_i^k \multimap \mathbf{PTM}_i^k$ in $\mathsf{STA}_\oplus$.

The initial configuration of a PTM is a configuration in the initial state $Q_0 = (q_1, \ldots, q_k)$ with the head at the beginning of a tape filled by 0's. Then, we need a term that, taking a numeral $\underline{n}_i$ as input, gives the encoding of the initial configuration with tape of length n as output.

Definition 60 (Indexed initial configuration)**.** For all $i, k \geq 1$, the *indexed initial configuration* is defined as follows:

$$\mathtt{init}_i \triangleq \lambda n.\lambda !c.\langle \lambda z.z, \lambda z.n\, !^i(\mathtt{d}^i(c)\, \underline{0})z, \underline{Q_0}\rangle.$$

Proposition 88. *For all $i, k \geq 1$, $\vdash_{\mathsf{STA}_\oplus} \mathtt{init}_i : \mathbf{N}_i \multimap \mathbf{PTM}^k_i$.*

The PTM needs now to be initialized with the given input string, by writing it on its tape. The term representing the initialization requires the term $\mathtt{decom}_i$ in (4.21).

Definition 61 (Indexed initialization)**.** For all $i, k \geq 1$, the *indexed initialization* is defined by $\mathtt{in}_i \triangleq \lambda s.\lambda m.s\, !(\lambda b.Tb \circ \mathtt{decom}_i)m$, where:

$$\begin{aligned}
T &\triangleq \lambda b.\lambda m.\lambda !c.\mathtt{let}\ m\ (!^i\mathtt{d}^i(c))\ \mathtt{be}\ l, r, c_l, b_l, c_r, b_r, q\ \mathtt{in}\\
&\qquad \mathtt{let}\ \mathtt{E}_{\mathbf{B}}\, b_r\ \mathtt{be}\ \mathbf{I}\ \mathtt{in}\ Rbq\langle l, r, c_l, b_l, c_r\rangle\\
R &\triangleq \lambda b'.\lambda q'.\lambda p.\mathtt{let}\ p\ \mathtt{be}\ l, r, c_l, b_l, c_r\ \mathtt{in}\ \langle c_r\, b' \circ c_l\, b_l \circ l, r, q'\rangle.
\end{aligned}$$

where $\mathtt{E}_{\mathbf{B}}$ is as in (3.3).

Proposition 89. *For all $i, k \geq 1$, $\vdash_{\mathsf{STA}_\oplus} \mathtt{in}_i : \mathbf{S} \multimap \mathbf{PTM}^k_i \multimap \mathbf{PTM}^k_i$.*

Last, we need to extract the output string from the final configuration.

Definition 62 (Indexed extraction)**.** For all $i, k \geq 1$, we define the *indexed extraction* as the following term:

$$\mathtt{ext}^{\mathbf{S}}_i \triangleq \lambda m.\lambda !c.\mathtt{let}\ m\, !^i(\mathtt{d}^i(c))\ \mathtt{be}\ l, r, q\ \mathtt{in}\ (\mathtt{let}\ \mathtt{E}_{\mathbf{B}^k}\, q\ \mathtt{be}\ I\ \mathtt{in}\ l \circ r).$$

where $\mathtt{E}_{\mathbf{B}^k}$ is an eraser of the ground type $\mathbf{B}^k$, that exists by Theorem 9.

Proposition 90. *For all $i, k \geq 1$, $\vdash_{\mathsf{STA}_\oplus} \mathtt{ext}^{\mathbf{S}}_i : \mathbf{PTM}^k_i \multimap \mathbf{S}_i$.*

By putting everything together, we are now able to encode a polytime PTM in $\mathsf{STA}_\oplus$.

Theorem 91 (Polytime completeness)**.** *Let $\mathcal{P}$ be a* PTM *such that:*

- *$\mathcal{P}$ runs in $p(n)$-time, for some polynomial $p : \mathbb{N} \longrightarrow \mathbb{N}$ with $\deg(p) = d_1$;*
- *$\mathcal{P}$ runs in $q(n)$-space, for some polynomial $q : \mathbb{N} \longrightarrow \mathbb{N}$ with $\deg(q) = d_2$;*
- *$\forall s \in \{0,1\}^*$, $\mathscr{L}_s : \{0,1\}^* \to [0,1]$ is the probabilistic distribution of the output strings of $\mathcal{P}$ on input s.*

Then there exists a term $\underline{\mathcal{P}}$ with type $!^{\max(d_1,d_2,1)+1}\mathbf{S} \multimap \mathbf{S}_{2d_2+1}$ in $\mathsf{STA}_\oplus$ such that, for every $s \in \{0,1\}^*$, there exists a surface distribution $\mathscr{D}_s$ satisfying the following conditions:

(1) $\underline{\mathcal{P}}\,(!^{\max(d_1,d_2,1)+1}\underline{s}) \Rightarrow \mathscr{D}_s$;

(2) $\forall s' \in \{0,1\}^*$, $\mathscr{D}_s(\underline{s'}) = \mathscr{L}_s(s')$.

Proof. Let $\mathcal{P}$ be a PTM running in polynomial time $p : \mathbb{N} \longrightarrow \mathbb{N}$ and in polynomial space $q : \mathbb{N} \longrightarrow \mathbb{N}$, with $\deg(p) = d_1$ and $\deg(q) = d_2$. We set $[p] = 2d_1 + 1$ and $[q] = 2d_2 + 1$. By Theorem 80 and Lemma 71 we have that the following judgements are derivable in $\mathsf{STA}_\oplus$:

$$\begin{aligned}
s_p : !^{d_1}\mathbf{S} &\vdash P : \mathbf{N}_{[p]}\\
s_q : !^{d_2}\mathbf{S} &\vdash Q : \mathbf{N}_{[q]}
\end{aligned} \tag{4.23}$$

where $P \triangleq \underline{p}\{!^{d_1}(\mathtt{len}_1\,\mathtt{d}^{d_1}(s_p))/x\}$, $Q \triangleq \underline{q}\{!^{d_2}(\mathtt{len}_1\,\mathtt{d}^{d_2}(s_q))/x\}$, and $\mathtt{len}_1$ is defined in (4.20). Again, by repeatedly applying Lemma 71 we can compose the terms in Definitions 59, 60, 61, and 62 to obtain a derivation in $\mathsf{STA}_\oplus$ of the following judgement:

$$s' : \mathbf{S}, p : \mathbf{N}_{[p]}, q : \mathbf{N}_{[q]} \vdash \mathtt{ext}^{\mathbf{S}}_{[q]}(p\,(!^{[p]}\mathtt{tr}_{[q]})(\mathtt{in}_{[q]}\,s'\,(\mathtt{init}_{[q]}\,q))) : \mathbf{S}_{[q]}. \tag{4.24}$$

By two further applications of Lemma 71, we can compose (4.23) and (4.24) to obtain the following:

$$s' : \mathbf{S}, s_p : !^{d_1}\mathbf{S}, s_q : !^{d_2}\mathbf{S} \vdash \mathtt{ext}^{\mathbf{S}}_{[q]}(P\,(!^{[p]}\mathtt{tr}_{[q]})(\mathtt{in}_{[q]}\,s'\,(\mathtt{init}_{[q]}\,Q))) : \mathbf{S}_{2d_2+1}.$$

By repeatedly applying rule m, and by applying rule $\multimap$I, we obtain the term:

$$\vdash_{\mathsf{STA}_\oplus} \underline{\mathcal{P}} : !^{\max(d_1,d_2,1)+1}\mathbf{S} \multimap \mathbf{S}_{2d_2+1}.$$

One can check that both point (1) and point (2) hold. □

4.4.3 Characterizing probabilistic complexity classes

In the previous subsection, $\mathsf{STA}_\oplus$ has been proved complete with respect to all polytime PTMs returning strings. But which complexity class our system is actually able to capture?

We start recalling some basic definitions from Arora and Barak [5].

Definition 63 (Recognising a language with error)**.** Let $\epsilon \in [0,1]$. Let $\mathcal{P}$ be a PTM and $L \subseteq \{0,1\}^*$ a language. We say that $\mathcal{P}$ *recognises L with error probability ϵ* if:

- $x \in L$ implies $\Pr[\mathcal{P}$ accepts $x] \geq 1-\epsilon$,
- $x \notin L$ implies $\Pr[\mathcal{P}$ rejects $x] \geq 1-\epsilon$,

where $\Pr[\mathcal{P}$ accepts $x]$ (resp. $\Pr[\mathcal{P}$ rejects $x]$) is the probability that $\mathcal{P}$ on input x terminates on an accepting (resp. rejecting) state. Moreover, if $T : \mathbb{N} \longrightarrow \mathbb{N}$ is a function, we say that $\mathcal{P}$ *recognizes L with error probability ϵ in $T(n)$-time* if it recognises L with error probability ϵ and, on every input x, it requires at most $T(|x|)$ steps of computation regardless of its random choices.

As opposed to the deterministic case, there are several probabilistic polytime complexity classes depending on the degree of accuracy we are willing to impose in recognising a language. We shall consider the classes PP (*Probabilistic Polynomial time*) and BPP (*Bounded-error Probabilistic Polynomial time*).

Definition 64 (The classes PP and BPP)**.**

- PP is the set of all languages that can be recognised by a PTM with error probability $0 \leq \epsilon \leq \frac{1}{2}$ in $p(n)$-time, for some polynomial $p : \mathbb{N} \longrightarrow \mathbb{N}$.
- BPP is the set of all languages that can be recognised by a PTM with error probability $0 \leq \epsilon < \frac{1}{2}$ in $p(n)$-time, for some polynomial $p : \mathbb{N} \longrightarrow \mathbb{N}$.

Observe that ϵ can be even equal to $\frac{1}{2}$ in PP, while it cannot in BPP. Due to this restriction, BPP enjoys an "amplification lemma", that gives a simple way of making the error probability exponentially small:

Lemma 92 (Amplification [86])**.** *Let $0 \leq \epsilon < \frac{1}{2}$. Then, for every polynomial $p : \mathbb{N} \longrightarrow \mathbb{N}$, a polytime PTM that recognises a language $L \subseteq \{0,1\}^*$ with error probability ϵ has an equivalent polytime PTM that recognises L with an error probability $2^{-p(n)}$.*

Several interesting languages in BPP have very efficient algorithms that recognise them. So, it is believed that BPP captures efficient probabilistic computation.

The following hierarchy is a straightforward consequence of Definition 64 (see Section 2.3.2):

Proposition 93 (Hierarchy)**.** $\mathsf{PTIME} \subseteq \mathsf{BPP} \subseteq \mathsf{PP}$.

It is still unknown if there are strict inclusions between these classes. A central open question of complexity theory is whether or not $\mathsf{BPP} = \mathsf{PTIME}$. Many complexity theorists believe that this equation holds, i.e. that there is a way to transform every probabilistic algorithm to a deterministic algorithm (one that does not toss any coin) while incurring only a polynomial slowdown.

Remark 15. Though very natural, BPP behaves differently from other classes. For example, PTIME and PP are often called *syntactic classes*, meaning that there is an easy way of checking if a machine recognises one of their languages. For example, every polytime PTM $\mathcal{P}$ recognises a language $L_{\mathcal{P}}$ in PP, which can be identified as follows. For all $x \in \{0,1\}^*$:

- if $\Pr[\mathcal{P} \text{ accepts } x] \geq \frac{1}{2}$ then $x \in L_{\mathcal{P}}$;
- otherwise, $\Pr[\mathcal{P} \text{ accepts } x] \leq \frac{1}{2}$, so $\Pr[\mathcal{P} \text{ reject } x] \geq \frac{1}{2}$ and $x \notin L_{\mathcal{P}}$.

This is not the case for BPP. Indeed, suppose that $\Pr[\mathcal{P} \text{ accepts } x] = \Pr[\mathcal{P} \text{ rejects } x] = \frac{1}{2}$, for some $x \in \{0,1\}^*$. Is $\mathcal{P}$ recognising a language L in BPP? No, because neither $x \in L$ nor $x \notin L$ hold, according to Definition 64. Due to these features, BPP is often called a *semantic class*.

We now need to define when a term typable in $\mathsf{STA}_{\oplus}$ recognises a language. A first approach is to let the bound on the error probability occur explicitly:

Definition 65 (Recognising a language with error ϵ)**.** Let $\epsilon \in [0,1]$, let $L \subseteq \{0,1\}^*$ be a language, and let M be a term with type $!^n\mathbf{S} \multimap \mathbf{B}$ in $\mathsf{STA}_{\oplus}$, for some $n \in \mathbb{N}$. We say that M *recognises* L *with error probability* ϵ if and only if the following conditions hold:

(1) if $s \in L$ and $M(!^n\underline{s}) \Rightarrow \mathscr{D}$, then $\mathscr{D}(\underline{0}) > 1 - \epsilon$;

(2) if $\underline{s} \notin L$ and $M(!^n\underline{s}) \Rightarrow \mathscr{D}$, then $\mathscr{D}(\underline{1}) > 1 - \epsilon$.

$\mathsf{STA}_{\oplus}$ captures both complexity classes. To show this, it suffices to replace the term $\mathtt{ext}_i^{\mathbf{S}}$ in the encoding of Theorem 91, which we recall extracts the output string from the final configuration of a PTM, with a term extracting the final state of a PTM (which is always accepting or rejecting by assumption). The latter term can be defined, for all $i, k \geq 1$, as follows:

$$\begin{aligned}\mathtt{ext}_i^{\mathbf{B}} \triangleq \lambda m.\mathtt{let}\ m\,!^i(\lambda b.\lambda c.\mathtt{let}\ \mathtt{E}_{\mathbf{B}}\,b\ \mathtt{be}\ I\ \mathtt{in}\ c)\ \mathtt{be}\ l, r, q \\ \mathtt{in}\ (l \circ r)(\underline{f}q) : \mathbf{PTM}_i^k \multimap \mathbf{B}\end{aligned}$$

where $\mathtt{E}_{\mathbf{B}}$ is as in (3.3) and f is a function deciding if a state is accepting or rejecting, that always exists by Lemma 81. The resulting encoding is able to represent all PTMs that, when fed with an input string, run in polynomial time returning the acceptance of not of a final configuration. Hence $\mathsf{STA}_{\oplus}$ is able to represent all PTMs recognising a language in PP or BPP:

Theorem 94 (PP and BPP)**.** *The set of languages which can be recognised with error ϵ in* $\mathsf{STA}_{\oplus}$ *for some $0 < \epsilon \leq \frac{1}{2}$ equals* PP. *The set of languages which can be recognised with error ϵ in* $\mathsf{STA}_{\oplus}$ *for some $0 < \epsilon < \frac{1}{2}$ equals* BPP.

In other words, $\mathsf{STA}_\oplus$ does not capture a single polytime complexity class, but several, depending on how the error bound has been set. Here comes the main drawback of Theorem 94, as observed in Dal Lago and Toldin [28]: the presence of an explicit, external error makes the characterization "less implicit", i.e. not in the style of ICC. A more implicit notion of characterization can be introduced by considering the so-called "representability by majority":

Definition 66 (Representability by majority)**.** Let $L \subseteq \{0,1\}^*$ be a language, and let M be a term with type $!^n\mathbf{S} \multimap \mathbf{B}$ in $\mathsf{STA}_\oplus$, for some $n \in \mathbb{N}$. We say that M *represents* L *by majority* if and only if the following conditions hold:

(1) if $s \in L$ and $M(!^n\underline{s}) \Rightarrow \mathscr{D}$, then $\mathscr{D}(\underline{0}) \geq \mathscr{D}(\underline{1})$;

(2) if $\underline{s} \notin L$ and $M(!^n\underline{s}) \Rightarrow \mathscr{D}$, then $\mathscr{D}(\underline{1}) \geq \mathscr{D}(\underline{0})$.

According to the above definition, $\mathsf{STA}_\oplus$ captures the class PP:

Theorem 95 (Completeness by majority for PP)**.** *The set of languages which can be represented by majority in* $\mathsf{STA}_\oplus$ *equals* PP.

No similar characterization is known for BPP. This fact should not be surprising, since BPP is a *semantic class*, as discussed in Remark 15: there is no easy way of deciding if a given PTM recognises a language in BPP. Now, if there were a "truly" implicit characterization in $\mathsf{STA}_\oplus$ of this class, listing all theorems of the system would provide a straightforward recursive enumeration of the PTMs recognising languages in BPP.

Chapter 5

The Benefit of Being Non-Lazy in Probabilistic λ-calculus

The probabilistic λ-calculus $\Lambda_\oplus$ extends the pure untyped λ-calculus with a sum $M \oplus N$, evaluating to M or N with equal probability 0.5. An operational semantics for $\Lambda_\oplus$ gives a function mapping a term M to a probability distribution $[\![M]\!]$ of values. As in the standard λ-calculus, different design choices may affect the meaning $[\![M]\!]$ of a term.

First, one has to decide how to evaluate a *β-redex*, i.e. the application of a function $\lambda x.M$ to an argument N. There are two main evaluation mechanisms: the *call-by-value policy* (cbv) consists in first evaluating N to some value V, then replacing the parameter x in M with V, while the *call-by-name policy* (cbn) replaces x with N as it is, before any evaluation. It is well-known that the two policies give rise to different results, especially in a probabilistic setting. Consider for example the term $(\lambda vz.vv)(\mathbf{T}\oplus\mathbf{F})$, where $\mathbf{T} = \lambda xy.x$ and $\mathbf{F} = \lambda xy.y$. In cbv, we first evaluate $\mathbf{T}\oplus\mathbf{F}$, yielding either $\mathbf{T}$ or $\mathbf{F}$ with equal probability, and then we pass the result to the function $\lambda vz.vv$, producing either $\lambda z.\mathbf{TT}$ or $\lambda z.\mathbf{FF}$, both with probability 0.5. By contrast, in cbn we pass the whole term $\mathbf{T}\oplus\mathbf{F}$ to the function before evaluating it, obtaining $\lambda z.(\mathbf{T}\oplus\mathbf{F})(\mathbf{T}\oplus\mathbf{F})$ with probability 1.

Second, one has to define which redexes to evaluate in a term, if any. Here again, there are various choices in λ-calculus: the *lazy strategy*, forbidding any reduction in the body of a function, so that $\lambda x.M$ is a value whatever M is, or the *head reduction*, consisting in reducing the redex in head position, which is at the left of any application. Concerning the lazy strategies, the meaning of a term is a distribution of weak head normal forms, i.e. either head normal forms without external abstractions or terms with form $\lambda x.M$. Concerning the head reduction, the meaning of a term is a distribution of head normal forms.

Let us remark that some variants of the standard head reduction have been studied, as for example the *head spine reduction* [84] that, given a β-redex $(\lambda x.M)N$, first evaluates the body of M and then evaluates the outermost redex according to cbn. One of the results of this chapter is that the head and head spine strategies are actually equivalent in (both the deterministic and) the probabilistic setting (Theorem 115). To the best of our knowledge, this result is not in the literature, even in the deterministic case.

Comparing terms by their operational semantics leads to undesired consequences, as higher-order normal forms differ often by syntactical details that are inessential with respect to their computational behaviour. Context equivalence is usually considered: two terms M, N are context equivalent ($M =_{\text{cxt}} N$ in symbols) whenever they “behave” the same in any possible “programming context”. In $\Lambda_\oplus$, a *context* $\mathcal{C}$ is a term with a special variable $[\cdot]$, the *hole*, and what we observe is the total mass of the distribution $[\![\mathcal{C}[M]]\!]$, i.e. the total probability of getting a result from the evaluation of the term $\mathcal{C}[M]$ obtained by replacing the hole with M. The definition of $=_{\text{cxt}}$ depends therefore on the chosen operational semantics.

Proving that two terms are context equivalent is rather difficult since we have to consider *all* contexts, hence the quest for more tractable equivalences comparable with $=_{\text{cxt}}$. We say in particular that an equivalence $\equiv$ over λ-terms is *sound* with respect to $=_{\text{cxt}}$ whenever the former implies the latter (i.e. $\equiv \subseteq =_{\text{cxt}}$), it is *complete* if the converse holds (i.e. $=_{\text{cxt}} \subseteq \equiv$) and it is *fully abstract* if it is both sound and complete, i.e. the two relations coincide.

In probabilistic λ-calculus, the first results in this line of research have been achieved in the setting of the denotational semantics of the $\Lambda_{\oplus}$ head reduction. In particular, Ehrhard et al. prove that the equivalence $\equiv_{\mathcal{D}^\infty}$ induced by the reflexive object $\mathcal{D}^\infty$ of the cartesian closed category of probabilistic coherence spaces [34] (as well as of the weighted relations [57]) is sound. More recently, Leventis proves a fundamental separation theorem, giving as a consequence that the probabilistic Nakajima tree equality is complete [60]. From the latter result, Clairambault and Paquet derive a fully abstract game model of $\Lambda_{\oplus}$ and as a corollary also the full abstraction of $\mathcal{D}^\infty$ [20]. The latter result has been also achieved independently by Leventis and Pagani [61].

All the above results deal with the head reduction, i.e. a non-lazy cbn operational semantics. For lazy strategies, a different approach is available, based on the notion of *applicative bisimulation*, which is the main subject of this chapter. The idea dates back to Abramsky [1] and consists in looking at the operational semantics as a transition system having λ-terms as states and transitions given by the evaluation of the application between λ-terms. The benefit of this setting is to transport into λ-calculus the whole theory of bisimilarity and its associated coinductive reasoning, which is a fundamental tool for comparing processes in concurrency theory. Basically, two terms M and N are applicative bisimilar (in symbols $M \sim N$) whenever their applications MP and NP reduce to applicative bisimilar values for any argument P.

This approach has been lifted to the probabilistic λ-calculus in a series of works by Dal Lago et al. [27, 22, 23], introducing the notion of *probabilistic applicative bisimilarity* (PAB) for lazy semantics. In particular, PAB is proven to be sound with the context equivalence in both cbv and cbn, but only cbv PAB is fully abstract. In case of lazy cbn, we have terms like:

$$M \triangleq \lambda xy.(x \oplus y) \qquad\qquad N \triangleq (\lambda xy.x) \oplus (\lambda xy.y) \tag{5.1}$$

such that $M =_{\text{cxt}} N$ but $M \not\sim N$. In fact, lazy PAB is able to discriminate between a term where a choice can be performed *before* any interaction, like N, and a term that needs to interact in order to trigger a choice, like M. Notice that this difference is caught also by cbv context semantics. For example, the two terms in (5.1) are distinguished by the context $\mathcal{C} = (\lambda v.(v\mathbf{I\Omega})(v\mathbf{I\Omega}))[\cdot]$ in cbv, because the total mass of $[\![\mathcal{C}[M]]\!]_{\text{cbv}}$ is 0.25, while that of $[\![\mathcal{C}[N]]\!]_{\text{cbv}}$ is 0.5. This is not the case in cbn, because $[\![\mathcal{C}[M]]\!]_{\text{cbn}} = [\![\mathcal{C}[N]]\!]_{\text{cbn}}$ has mass 0.25.

In [27] the authors analyse the above example remarking that the cbn policy misses the "capability of copying a term *after* having evaluated it". This is indeed a fundamental primitive in probabilistic programming: when implementing a probabilistic algorithm we need often to toss a coin and then to pass *the result* of this tossing to several subroutines. It is then common to extend a probabilistic language with a `let` constructor, often called *sampling*, evaluating a choice *before* passing it to a function even in a cbn semantics. As expected, it is shown by Kasterovic and Pagani [53] that such an extension recovers cbn PAB full abstraction, as terms like (5.1) become contextually different.

The above considerations lead us to the following striking observations. First, it has been proven that in simply typed languages the presence of the `let` constructor does not affect the discriminating power of the context equivalence. For example, in probabilistic PCF the lazy cbn context equivalence coincides with the equality in the model of probabilistic coherence spaces [36, 37], with or without a sampling primitive. Why this neat difference with an untyped framework? Second, we have already mentioned several denotational models of $\Lambda_{\oplus}$ which are fully abstract with respect to a pure cbn context equivalence, so without this "capability of copying a term *after* having evaluated it". Is it really so necessary for getting a fully abstract PAB?

The first question can be easily answered by focussing on the laziness constraint of the operational semantics. Every λ-abstraction is a value for a lazy semantics. This does not affect the set of observables in a simply typed setting (as PCF), because it is defined on ground types (booleans, numerals, etc). By contrast, every term is a function in an untyped setting, so the laziness radically changes what we can observe in the behaviour of a term. The goal of this chapter is to show that also the second question deals with laziness: we prove that PAB is fully abstract for the head reduction (Theorem 134). This is unexpected: non-lazy semantics seems to have no need of the sampling primitives in order to have fully abstract PAB, even with a cbn policy and an untyped setting.

On a more technical side, we stress that our proofs of soundness and completeness follow a different reasoning than the one used in probabilistic lazy semantics [22, 23, 53]. First, the soundness ($\sim \subseteq =_{\mathrm{cxt}}$) does not need a Howe lifting [52], as we prove a Context Lemma (Lemma 119) for $=_{\mathrm{cxt}}$ and an applicative property of $\sim$ (Lemma 125), the latter using the notion of probabilistic assignments as in [27]. Second, and more fundamental, the proof of completeness ($=_{\mathrm{cxt}} \subseteq \sim$) is not achieved by transforming PAB into a testing equivalence using a theorem by van Breugel et al. [92], as in Crubillé and Dal Lago [22]. Rather, we use the Leventis Separation property [60] to prove that the context equivalence is a probabilistic applicative bisimulation and so contained in PAB by definition (Theorem 134).

What about inequalities? All equivalences introduced so far have an asymmetric version: the context preorder and the probabilistic applicative similarity (PAS). We prove also that PAS is sound but not complete with respect to the context inequality. We give a counterexample to the full abstraction in the asymmetric case (see (5.16)) and we argue that extending the calculus with Plotkin's parallel disjunction [76], as done by Crubillé and Dal Lago in [23], is enough to circumvent the counterexample. This leaves some room for the conjecture that the full abstraction for PAS can be somehow restored in this extended calculus.

Outline of the chapter. In this chapter we show that probabilistic applicative bisimilarity is fully abstract for the head reduction, also known as non-lazy cbn, in the pure and untyped probabilistic λ-calculus. In Section 5.1 we present the probabilistic λ-calculus endowed with a big-step probabilistic operational semantics based on the head *spine* reduction (Section 5.1.1 and 5.1.2), and we define the context preorder and context equivalence relations (Section 5.1.3). Then we recall the basic notions and results about probabilistic similarity and bisimilarity (Section 5.1.4), and we introduce PAS (probabilistic applicative similarity) and PAB (probabilistic applicative bisimilarity) (Section 5.1.5). In Section 5.2 we prove that the head reduction and the head spine reduction yield the same operational semantics. On the one hand, the latter evaluation policy allows for a simpler proof of the Soundness Theorem. On the other hand, the equivalence between the two reduction strategies enables us to show the Completeness Theorem using Leventis' Separation [60] for the head reduction. In Section 5.3 we prove the Context Lemma (Section 5.3.1), and we apply it to show that PAS is included in the context preorder relation, and hence that PAB is sound with respect to the context equivalence (Section 5.3.2). In Section 5.3 we briefly recall from [60] the probabilistic Nakajima trees and the Separation Theorem (Section 5.4.1), and we use the latter to prove that PAB is complete, and hence fully abstract, with respect to the context equivalence (Section 5.4.2). Finally, we introduce a counterexample to the completeness property relating PAS with the context preorder (Section 5.4.3), and we discuss how a full abstraction result for PAS can be obtained in an extended calculus

(Section 5.4.4).

5.1 Preliminaries

In this section we introduce the fundamental notions of the chapter. We first present the syntax and the operational semantics of the probabilistic λ-calculus $\Lambda_\oplus$, on top of which we shall consider the context preorder and the context equivalence relations. Then, we recall Larsen and Skou's probabilistic (bi)similarity on labelled Markov chains [58]. Following [27, 22, 53], we shall apply Abramsky's applicative (bi)similarity [1] to the operational semantics of $\Lambda_\oplus$, getting the probabilistic applicative (bi)similarity.

5.1.1 The probabilistic λ-calculus $\Lambda_\oplus$

The probabilistic λ-calculus is the pure, untyped λ-calculus extended with a binary sum operator $\oplus$ representing a fair choice. The terms of the probabilistic λ-calculus are defined like in [27, 53], where the head normal forms will be considered as values:

Definition 67 (Terms and head normal forms)**.** Let $\mathcal{V} = \{x, y, \ldots\}$ be a denumerable set of variables. The set $\Lambda_\oplus$ of *(probabilistic λ-)terms* is generated by the following grammar:

$$M := x \mid \lambda x.M \mid MM \mid M \oplus M \tag{5.2}$$

where $x \in \mathcal{V}$. A term is in (or is a) *head normal form* if it is of the form $\lambda x_1 \ldots x_n.yN_1 \ldots N_m$, for some $n, m \in \mathbb{N}$. If $n = 0$ then the term is also called *neutral*. Head normal forms are ranged over by metavariables like H. The set of all head normal forms will be denoted by HNF, the set of all neutral terms will be denoted by NEUT.

Terms are considered modulo renaming of bound variables. The set of free variables of a term M (i.e. $FV(M)$) and the clash-free substitution of N for the free occurrences of x in M (i.e. $M[N/x]$) are defined in a standard way. Finite subsets of $\mathcal{V}$ are ranged over by Γ. Given Γ, the set of terms (resp. head normal forms) whose free variables are within Γ is denoted $\Lambda_\oplus^\Gamma$ (resp. HNF^Γ).

Example 22. Useful terms are the identity $\mathbf{I} \triangleq \lambda x.x$, the boolean values $\mathbf{T} \triangleq \lambda xy.x$ and $\mathbf{F} \triangleq \lambda xy.y$, the duplicator $\boldsymbol{\Delta} \triangleq \lambda x.xx$, the ever looping term $\boldsymbol{\Omega} \triangleq \boldsymbol{\Delta\Delta}$ and the Turing fixed-point combinator $\boldsymbol{\Theta} \triangleq (\lambda x.\lambda y.(y(xxy)))(\lambda x.\lambda y.(y(xxy)))$. An example of probabilistic λ-term that does not belong to the standard λ-calculus is $\mathsf{hid} \triangleq \mathbf{I} \oplus \boldsymbol{\Omega}$.

Definition 68 (Context)**.** A *context* of $\Lambda_\oplus$ is a term containing a unique hole $[\cdot]$, generated by the following grammar:

$$\mathcal{C} := [\cdot] \mid \lambda x.\mathcal{C} \mid \mathcal{C}M \mid M\mathcal{C} \mid \mathcal{C} \oplus M \mid M \oplus \mathcal{C} \tag{5.3}$$

We denote by $\mathsf{C}\Lambda_\oplus$ the set of all contexts. Given $\mathcal{C} \in \mathsf{C}\Lambda_\oplus$ and $M \in \Lambda_\oplus$, $\mathcal{C}[M]$ denotes a term obtained by substituting the unique hole in $\mathcal{C}$ with M allowing the possible capture of free variables of M. A *head context* is a context of the form $\lambda x_1 \ldots x_n.[\cdot]L_1 \ldots L_m$, or $\lambda \vec{x}.[\cdot]\vec{L}$ for short, where $n, m \geq 0$ and $L_i \in \Lambda_\oplus$. We denote by $\mathsf{H}\Lambda_\oplus$ the set of all head contexts, that are ranged over by $\mathcal{E}$.

In the probabilistic semantics for $\Lambda_\oplus$, a term reduces not to a single head normal form but rather to a subprobability distribution over HNF:

$$\frac{}{M \Downarrow \bot}\, s1 \qquad \frac{}{x \Downarrow x}\, s2 \qquad \frac{M \Downarrow \mathscr{D}}{\lambda x.M \Downarrow \lambda x.\mathscr{D}}\, s3$$

$$\frac{M \Downarrow \mathscr{D} \qquad \{H[N/x] \Downarrow \mathscr{E}_{H,N}\}_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{D})}}{MN \Downarrow \displaystyle\sum_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{D})} \mathscr{D}(\lambda x.H) \cdot \mathscr{E}_{H,N} + \sum_{H \,\in\, \mathrm{supp}(\mathscr{D}) \,\cap\, \mathrm{NEUT}} \mathscr{D}(H) \cdot HN}\, s4$$

$$\frac{M \Downarrow \mathscr{D} \qquad N \Downarrow \mathscr{E}}{M \oplus N \Downarrow \frac{1}{2} \cdot \mathscr{D} + \frac{1}{2} \cdot \mathscr{E}}\, s5$$

Figure 5.1: Big-step approximation.

Definition 69 (Head distributions)**.** A *head distribution* is a subprobability distribution over HNF, i.e. a function $\mathscr{D} : \mathrm{HNF} \longrightarrow [0, 1]$ such that:

$$\sum_{H \in \mathrm{HNF}} \mathscr{D}(H) \leq 1.$$

In Section 2.5.2 we introduced the basic definitions and conventions related to subprobability distributions. So, for example, $\mathfrak{D}(\mathrm{HNF})$ stands for the set of all head distributions and $\leq_{\mathfrak{D}}$ for the pointwise order on $\mathfrak{D}(\mathrm{HNF})$; moreover, $\bot$ denotes the null distribution, while $\mathscr{D}(X)$ denotes $\sum_{H \in X} \mathscr{D}(H)$ whenever $X \subseteq \mathrm{HNF}$. We may also write $\mathscr{D}(X)$ for a generic subset $X \subseteq \Lambda_{\oplus}$ of terms, meaning in fact $\mathscr{D}(X \cap \mathrm{HNF})$.

Head distributions are *subprobability* distributions, and hence they do not necessarily sum to 1. This allows us to model divergence, and to look at some distributions as "approximations" of others by comparing them with the pointwise order $\leq_{\mathfrak{D}}$ on $\mathfrak{D}(\mathrm{HNF})$, where $(\mathfrak{D}(\mathrm{HNF}), \leq_{\mathfrak{D}})$ is a directed-complete partial order with least element $\bot$ (see (2.2) in Section 2.5.2).

Given a head distribution $\mathscr{D}$, the head distribution $\lambda x.\mathscr{D}$ is defined, for all $H \in \mathrm{HNF}$, as follows:

$$(\lambda x.\mathscr{D})(H) \triangleq \begin{cases} \mathscr{D}(H') & \text{if } H = \lambda x.H', \text{ for some } H' \in \mathrm{HNF}, \\ 0 & \text{otherwise.} \end{cases}$$

We now endow $\Lambda_{\oplus}$ with a big-step probabilistic operational semantics based on the head *spine* reduction strategy [84], a variant of the usual head reduction. Following Dal Lago and Zorzi [29], this can be done in two stages. First we inductively define a notion of big-step approximation relation $M \Downarrow \mathscr{D}$ between a term M and head distribution $\mathscr{D}$, which captures convergence. Then, we define the big-step semantics $[\![M]\!]$ of M as the supremum of all its big-step approximations.

Definition 70 (Big-step approximation)**.** The big-step approximation relation $\Downarrow \subseteq \Lambda_{\oplus} \times \mathfrak{D}(\mathrm{HNF})$ is defined by the rules in Figure 5.1.

The approximation relation $\Downarrow$ in Definition 70 is not a function: many different head distributions can be put in correspondence with the same term M, because of the rule $s1$ that allows one to "give up" while looking for a distribution of a term. In other words, $\Downarrow$ is not meant to be a way to attribute *one* head distribution to every term, but rather to find all finitary approximants of the *unique* head distribution we are looking for.

Definition 71 (Big-step semantics)**.** The *big-step semantics* of a term $M \in \Lambda_\oplus$ is defined by:

$$[\![M]\!] \triangleq \sup\{\mathscr{D} \,|\, M \Downarrow \mathscr{D}\}.$$

The big-step semantics of a term is *always* a head distribution, as a consequence of (2.2) (Section 2.5.2) and the following lemma:

Lemma 96. *For every $M \in \Lambda_\oplus$, $\{\mathscr{D} \in \mathfrak{D}(\mathrm{HNF}) \mid M \Downarrow \mathscr{D}\}$ is a directed set.*

Proof. We have to show that, for every $M \in \Lambda_\oplus$, if $M \Downarrow \mathscr{D}$ and $M \Downarrow \mathscr{E}$ then there exits $\mathscr{F} \in \mathfrak{D}(\mathrm{HNF})$ such that $M \Downarrow \mathscr{F}$ and $\mathscr{D}, \mathscr{E} \leq_{\mathfrak{D}} \mathscr{F}$. The proof is by induction on the structure of the derivations of $M \Downarrow \mathscr{D}$ and $M \Downarrow \mathscr{E}$. If $\mathscr{D} = \bot$ then $\mathscr{F} \triangleq \mathscr{E}$. Similarly, if $\mathscr{E} = \bot$ then $\mathscr{F} \triangleq \mathscr{D}$. Otherwise, we consider the structure of M. If M is a variable, say x, then the last rule of both $M \Downarrow \mathscr{D}$ and $M \Downarrow \mathscr{E}$ is $s2$, and we set $\mathscr{F} \triangleq x$. If M is an abstraction, say $\lambda x.M'$, then the last rule of both $M \Downarrow \mathscr{D}$ and $M \Downarrow \mathscr{E}$ is $s3$:

$$\frac{M' \Downarrow \mathscr{D}'}{\lambda x.M' \Downarrow \mathscr{D}}\, s3 \qquad\qquad \frac{M' \Downarrow \mathscr{E}'}{\lambda x.M' \Downarrow \mathscr{E}}\, s3$$

By induction hypothesis, there exists $\mathscr{F}'$ such that $M' \Downarrow \mathscr{F}'$ and $\mathscr{D}', \mathscr{E}' \leq_{\mathfrak{D}} \mathscr{F}'$, so that we set $\mathscr{F} \triangleq \lambda x.\mathscr{F}'$. If M is an application, say $M'N$, the last rule of both $M \Downarrow \mathscr{D}$ and $M \Downarrow \mathscr{E}$ is $s4$:

$$\frac{M' \Downarrow \mathscr{D}' \qquad \{H[N/x] \Downarrow \mathscr{D}''_{H,N}\}_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{D}')}}{M'N \Downarrow \mathscr{D}}\, s4$$

$$\frac{M' \Downarrow \mathscr{E}' \qquad \{H[N/x] \Downarrow \mathscr{E}''_{H,N}\}_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{E}')}}{M'N \Downarrow \mathscr{E}}\, s4$$

By induction hypothesis, there exist $\mathscr{F}'$ such that $M' \Downarrow \mathscr{F}'$ and $\mathscr{D}', \mathscr{E}' \leq_{\mathfrak{D}} \mathscr{F}'$. Moreover, for all $H \in \mathrm{supp}(\mathscr{F}')$, if $H \in \mathrm{supp}(\mathscr{D}') \cap \mathrm{supp}(\mathscr{E}')$ then, by induction hypothesis, there exists $\mathscr{G}''_{H,N}$ such that $H[L/x] \Downarrow \mathscr{G}''_{H,N}$ and $\mathscr{D}''_{H,N}, \mathscr{E}''_{H,N} \leq \mathscr{G}''_{H,N}$. Hence, we set:

$$\mathscr{F}''_{H,N} \triangleq \begin{cases} \mathscr{D}''_{H,N} & \text{if } H \in \mathrm{supp}(\mathscr{D}') \text{ and } H \notin \mathrm{supp}(\mathscr{E}'), \\ \mathscr{E}''_{H,N} & \text{if } H \in \mathrm{supp}(\mathscr{E}') \text{ and } H \notin \mathrm{supp}(\mathscr{D}'), \\ \mathscr{G}''_{H,N} & \text{if } H \in \mathrm{supp}(\mathscr{D}') \cap \mathrm{supp}(\mathscr{E}'), \\ \bot & \text{otherwise.} \end{cases}$$

Then, we define:

$$\mathscr{F} \triangleq \sum_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{F}')} \mathscr{F}'(\lambda x.H) \cdot \mathscr{F}''_{H,N} + \sum_{H \in\, \mathrm{supp}(\mathscr{F}') \,\cap\, \mathrm{NEUT}} \mathscr{F}'(H) \cdot HN$$

The last case is when M is a probabilistic sum, say $M' \oplus M''$. Then the last rule of both $M \Downarrow \mathscr{D}$ and $M \Downarrow \mathscr{E}$ is $s5$:

$$\frac{M' \Downarrow \mathscr{D}' \qquad M'' \Downarrow \mathscr{D}''}{M' \oplus M'' \Downarrow \mathscr{D}}\, s5 \qquad\qquad \frac{M' \Downarrow \mathscr{E}' \qquad M'' \Downarrow \mathscr{E}''}{M' \oplus M'' \Downarrow \mathscr{E}}\, s5$$

By induction hypothesis, there exist $\mathscr{F}'$ and $\mathscr{F}''$ such that $M' \Downarrow \mathscr{F}'$ and $\mathscr{D}', \mathscr{E}' \leq_{\mathfrak{D}} \mathscr{F}'$, as well as $M'' \Downarrow \mathscr{F}''$ and $\mathscr{D}'', \mathscr{E}'' \leq_{\mathfrak{D}} \mathscr{F}''$. Then, it suffices to define $\mathscr{F} \triangleq \frac{1}{2} \cdot \mathscr{F}' + \frac{1}{2} \cdot \mathscr{F}''$. $\square$

Note that, if M is deterministic, i.e. a term without the probabilistic sum $\oplus$, then either M has a unique head normal form H and $[\![M]\!](H) = 1$, or M is a diverging term and $[\![M]\!] = \bot$. So $[\![\cdot]\!]$ generalises the usual deterministic semantics.

Example 23. Consider the term $M \triangleq \mathbf{\Delta}(\mathbf{T} \oplus \mathbf{F})$. One can easily check that the rules in Figure 5.1 allow us to derive $M \Downarrow \mathscr{D}$ for any $\mathscr{D}$ in the following set $\Big\{\bot,\ \frac{1}{4} \cdot \lambda y.\mathbf{T},\ \frac{1}{4} \cdot \lambda y.\mathbf{F},\ \frac{1}{2} \cdot \mathbf{I},\ \frac{1}{4} \cdot \lambda y.\mathbf{T} + \frac{1}{4} \cdot \lambda y.\mathbf{F},\ \frac{1}{4} \cdot \lambda y.\mathbf{T} + \frac{1}{2} \cdot \mathbf{I},\ \frac{1}{4} \cdot \lambda y.\mathbf{F} + \frac{1}{2} \cdot \mathbf{I},\ \frac{1}{4} \cdot \lambda y.\mathbf{T} + \frac{1}{4} \cdot \lambda y.\mathbf{F} + \frac{1}{2} \cdot \mathbf{I}\Big\}$. The latter head distribution is the supremum of this set and so it defines the semantics of M.

Example 23 is about normalizing terms, which means here terms M with semantics of total mass $\sum [\![M]\!] = 1$ and such that there exists a unique finite derivation giving $M \Downarrow [\![M]\!]$. Standard non-converging terms give partiality:

Example 24. By inspection on the rule s4 in Figure 5.1, one can check that $\mathbf{\Omega} \Downarrow \mathscr{D}$ only if $\mathscr{D} = \bot$, so $[\![\mathbf{\Omega}]\!] = \bot$. As a consequence we also have, e.g. $[\![\mathbf{\Omega} \oplus \mathbf{I}]\!] = \frac{1}{2} \cdot \mathbf{I}$.

The probabilistic λ-calculus allows us also for *almost sure terminating* terms, namely terms M such that $\sum [\![M]\!] = 1$ but *without* finite derivations of $M \Downarrow [\![M]\!]$:

Example 25. Consider the derivation of $MM \Downarrow \sum_{i=1}^{n} \frac{1}{2^i} \cdot y$ depicted in Figure 5.2, where $M \triangleq \lambda x.(y \oplus xx)$. Any such finite approximation of $[\![MM]\!]$ gives a head distribution of the form $\sum_{i=1}^{n} \frac{1}{2^i} \cdot y$, for some $n \geq 1$, but only the limit sum $\sup_{i=1}^{n} \sum \frac{1}{2^i} \cdot y$ is equal to y, thus yielding $[\![MM]\!] = y$.

The operational semantics can be defined inductively, as the following proposition states:

Proposition 97. *For every $M, N \in \Lambda_{\oplus}$ and $H \in \mathrm{HNF}$:*

(1) $[\![MN]\!] = \sum_{\lambda x.H \,\in\, \mathrm{supp}([\![M]\!])} [\![M]\!](\lambda x.H) \cdot [\![H[N/x]]\!] \ + \sum_{H \,\in\, \mathrm{supp}([\![M]\!]) \,\cap\, \mathrm{NEUT}} [\![M]\!](H) \cdot HN$*,*

(2) $[\![(\lambda x.H)N]\!] = [\![H[N/x]]\!]$*,*

(3) $[\![\lambda x.M]\!] = \lambda x.[\![M]\!]$*,*

(4) $[\![M \oplus N]\!] = \frac{1}{2}[\![M]\!] + \frac{1}{2}[\![N]\!]$*.*

Moreover, for every $H \in \mathrm{HNF}$, $[\![H]\!] = H$.

Proof. First, we prove point (1). Let $\mathscr{D}$ be such that $MN \Downarrow \mathscr{D}$. The case $\mathscr{D} = \bot$ is trivial, so suppose $\mathscr{D} \neq \bot$. Then, $MN \Downarrow \mathscr{D}$ must be obtained by applying the rule $s4$ to the premises $M \Downarrow \mathscr{E}$ and $\{H[N/x] \Downarrow \mathscr{F}_{H,N}\}_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{E})}$, so that $\mathscr{D}$ is of the form:

$$\sum_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{E})} \mathscr{E}(\lambda x.H) \cdot \mathscr{F}_{H,N} \ + \sum_{H \,\in\, \mathrm{supp}(\mathscr{E}) \,\cap\, \mathrm{NEUT}} \mathscr{E}(H) \cdot HN \tag{5.4}$$

This proves the $\leq_{\mathfrak{D}}$ direction. For the converse, suppose that $\mathscr{E}$ is a head distribution such that $M \Downarrow \mathscr{E}$ and, for all $\lambda x.H \in \mathrm{supp}(\mathscr{E})$, suppose $\mathscr{F}_{H,N}$ is a head distribution such that $H[N/x] \Downarrow \mathscr{F}_{H,N}$. By applying rule $s4$, we get $MN \Downarrow \mathscr{D}$, where $\mathscr{D}$ is as in (5.4), and the result follows.

Point (2) is a special case of point (1) where $M = \lambda x.H$. So, let us prove point (3). As for the $\leq_{\mathfrak{D}}$ direction, suppose $\lambda x.M \Downarrow \mathscr{D}$. The case $\mathscr{D} = \bot$ is trivial, so suppose $\mathscr{D} \neq \bot$. Then,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{y \Downarrow y}\,s2 \quad \dfrac{\dfrac{}{x \Downarrow x}\,s2}{xx \Downarrow xx}\,s4}{y \oplus xx \Downarrow \frac{1}{2} \cdot y + \frac{1}{2} \cdot xx}\,s5}{M \Downarrow \frac{1}{2} \cdot \lambda x.y + \frac{1}{2} \cdot \boldsymbol{\Delta}}\,s3 \quad \dfrac{}{y \Downarrow y}\,s2 \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y \Downarrow y}\,s2 \quad \dfrac{\dfrac{}{x \Downarrow x}\,s2}{xx \Downarrow xx}\,s4}{y \oplus xx \Downarrow \frac{1}{2} \cdot y + \frac{1}{2} \cdot xx}\,s5}{M \Downarrow \frac{1}{2} \cdot \lambda x.y + \frac{1}{2} \cdot \boldsymbol{\Delta}}\,s3 \quad \dfrac{}{y \Downarrow y}\,s2 \quad \begin{matrix} \dfrac{\dfrac{\dfrac{\dfrac{}{y \Downarrow y}\,s2 \quad \dfrac{\dfrac{}{x \Downarrow x}\,s2}{xx \Downarrow xx}\,s4}{y \oplus xx \Downarrow \frac{1}{2} \cdot y + \frac{1}{2} \cdot xx}\,s5}{M \Downarrow \frac{1}{2} \cdot \lambda x.y + \frac{1}{2} \cdot \boldsymbol{\Delta}}\,s3 \quad \dfrac{}{y \Downarrow y}\,s2 \quad \dfrac{}{MM \Downarrow \bot}\,s1}{MM \Downarrow \frac{1}{2} \cdot y}\,s4 \\ \vdots \end{matrix}}{MM \Downarrow \sum_{i=1}^{n-1} \frac{1}{2^i} \cdot y}\,s4}{MM \Downarrow \sum_{i=1}^{n} \frac{1}{2^i} \cdot y}\,s4$$

Figure 5.2: A derivation in the big-step semantics of $MM \Downarrow \sum_{i=1}^{n} \frac{1}{2^i} \cdot y$, where $M \triangleq \lambda x.(y \oplus xx)$ and $\boldsymbol{\Delta} = \lambda x.xx$.

$\lambda x.M \Downarrow \mathscr{D}$ must be obtained from $M \Downarrow \mathscr{D}'$ by applying rule $s3$, where $\mathscr{D} = \lambda x.\mathscr{D}'$, so that $[\![\lambda x.M]\!] \leq_{\mathfrak{D}} \lambda x.[\![M]\!]$. For the converse, suppose $\mathscr{D}$ is a head distribution such that $M \Downarrow \mathscr{D}$. By applying rule $s3$ we get $\lambda x.M \Downarrow \lambda x.\mathscr{D}$, so that $\lambda x.[\![M]\!] \leq_{\mathfrak{D}} [\![\lambda x.M]\!]$. Point (4) is similar.

Finally, for all $H \in \text{HNF}$, we prove $[\![H]\!] = H$ by induction on the structure of H. If H is a variable, say x, then $[\![x]\!] = x$. If H is an abstraction, say $\lambda x.H'$, then H' is a head normal form. By induction hypothesis, $[\![H']\!] = H'$. By point (3) we have $[\![\lambda x.H']\!] = \lambda x.[\![H']\!] = \lambda x.H'$. Last, if H is an application, say MN, then M must be of the form $xP_1 \dots P_n$. By point (1), we have $[\![MN]\!] = [\![xP_1 \dots P_n]\!](xP_1 \dots P_n) \cdot xP_1 \dots P_n N = xP_1 \dots P_n N$. □

5.1.2 Head reduction and head spine reduction

The rules in Figure 5.1 do not correspond to the standard head reduction of the λ-calculus, but implement a variant of it, called *head spine* reduction in [84]. Both the head and head spine reduction strategies can be introduced as probabilistic transition relations over $\Lambda_\oplus$, i.e. relations $\mathcal{R} \subseteq \Lambda_\oplus \times [0,1] \times \Lambda_\oplus$ such that, for all $M \in \Lambda_\oplus$, $\sum_{p,\, N \text{ s.t. } M\ \mathcal{R}_p\ N} p \leq 1$ (see Section 2.5.2).

Definition 72 (Head and head spine reductions)**.** We define $\to$ (*head reduction*) and $\dashrightarrow$ (*head spine reduction*) as the following probabilistic transition relations over $\Lambda_\oplus$:

$$M \to_p N \triangleq \begin{cases} M = \mathcal{E}[(\lambda y.P)Q],\ N = \mathcal{E}[P[Q/y]],\ \mathcal{E} \in \mathsf{H}\Lambda_\oplus,\ p = 1, \\ \qquad\qquad\qquad\qquad \text{or} \\ M = \mathcal{E}[P_1 \oplus P_2],\ P_1 \neq P_2,\ N = \mathcal{E}[P_i],\ \mathcal{E} \in \mathsf{H}\Lambda_\oplus,\ p = \frac{1}{2}, \\ \qquad\qquad\qquad\qquad \text{or} \\ M = \mathcal{E}[P \oplus P],\ N = \mathcal{E}[P],\ \mathcal{E} \in \mathsf{H}\Lambda_\oplus,\ p = 1. \end{cases}$$

$$M \dashrightarrow_p N \triangleq \begin{cases} M = \mathcal{E}[(\lambda y.H)Q],\ N = \mathcal{E}[H[Q/y]],\ H \in \text{HNF},\ \mathcal{E} \in \mathsf{H}\Lambda_\oplus,\ p = 1, \\ \qquad\qquad\qquad\qquad \text{or} \\ M = \mathcal{E}[(\lambda y.P)Q],\ P \dashrightarrow_p P',\ N = \mathcal{E}[(\lambda y.P')Q],\ \mathcal{E} \in \mathsf{H}\Lambda_\oplus, \\ \qquad\qquad\qquad\qquad \text{or} \\ M = \mathcal{E}[P_1 \oplus P_2],\ P_1 \neq P_2,\ N = \mathcal{E}[P_i],\ \mathcal{E} \in \mathsf{H}\Lambda_\oplus,\ p = \frac{1}{2}, \\ \qquad\qquad\qquad\qquad \text{or} \\ M = \mathcal{E}[P \oplus P],\ N = \mathcal{E}[P],\ \mathcal{E} \in \mathsf{H}\Lambda_\oplus,\ p = 1. \end{cases}$$

Let us state some remarkable properties concerning both the head and head spine reductions:

Lemma 98. *Let $M, N, L \in \Lambda_\oplus$:*

(1) Application: *if $M \dashrightarrow_p N$ then $ML \dashrightarrow_p NL$,*

(2) Substitution: *if $M \to_p N$ then $M[L/x] \to_p N[L/x]$,*

(3) Abstraction: *if $M\ \mathcal{R}_p\ N$ then $\lambda x.M\ \mathcal{R}_p\ \lambda x.N$, for $\mathcal{R} \in \{\to, \dashrightarrow\}$.*

Proof. Straightforward. □

Observe that the application property does not hold for the head reduction. For example, $\lambda x.\mathbf{II} \to_p \lambda x.\mathbf{I}$, but $(\lambda x.\mathbf{II})\mathbf{I} \to_p \mathbf{II} \neq (\lambda x.\mathbf{I})\mathbf{I}$. Also, the substitution property does not hold for the head spine reduction. For example, if $M \triangleq (\lambda x.y)\mathbf{I}$ then $M \dashrightarrow_p y$ but $M[\mathbf{\Omega}/y] \dashrightarrow_p M[\mathbf{\Omega}/y] \neq y[\mathbf{\Omega}/y]$.

Let us see the difference between the two reduction strategies on a deterministic λ-term, e.g. $M \triangleq (\lambda x.(\lambda y.x)y)z$. The (small-step) head reduction first evaluates the outermost redex of M, getting $(\lambda y.z)y$, and then the latter term, terminating in the head normal form z. The (small-step) head spine reduction first evaluates the body of $\lambda x.(\lambda y.x)y$ to a head normal form, so getting the term $\lambda x.x$ and then it fires the application of the latter to the variable z, getting z. The two reduction sequences are different but they give the same result (and actually with the same number of reduction steps). We prove in Theorem 111 that this is always the case, even in a probabilistic setting. Hence, the definition of $[\![\cdot]\!]$ in Definition 71 is just another way of presenting the operational semantics generated by the head reduction and discussed, for example, in [35, 60, 61].

We decided to consider the head spine reduction for several reasons. First, because it has a compact big-step presentation. Indeed, defining a big-step semantics based on the head reduction strategy requires a further notion of value as well as a further approximation relation. On the one hand, beside the head distributions $\mathscr{D}$, we need distributions $\mathscr{W}$ of *weak head normal forms*, i.e. terms which are either neutral or abstractions. On the other hand, beside the big-step approximation relation based on the head reduction strategy, say $\Downarrow_{\mathrm{h}}$, we need another one for the lazy call-by-name evaluation, we denote by $\downarrow_{\mathrm{cbn}}$. Then, for example, the big-step rule for the application looks like the following:

$$\frac{M \downarrow_{\mathrm{cbn}} \mathscr{W} \qquad \{P[N/x] \Downarrow_{\mathrm{h}} \mathscr{D}_{P,N}\}_{\lambda x.P \,\in\, \mathrm{supp}(\mathscr{W})}}{MN \Downarrow_{\mathrm{h}} \sum_{\lambda x.P \,\in\, \mathrm{supp}(\mathscr{W})} \mathscr{W}(\lambda x.P) \cdot \mathscr{D}_{P,N} + \sum_{H \,\in\, \mathrm{supp}(\mathscr{W}) \,\cap\, \mathrm{NEUT}} \mathscr{W}(H) \cdot HN}$$

The operational meaning of the above rule can be described as follows: whenever an application MN is reached during the head reduction, M needs to be evaluated under a lazy call-by-name policy until some weak head normal form is obtained; if the weak head normal form is an abstraction $\lambda x.P$, then we apply the head reduction on $P[N/x]$.

Another reason why we introduced an operational semantics based on the head spine reduction is because it fits perfectly into the $\Lambda_\oplus$-Markov chain definition, as we shall see in Remark 16. On the one side, this allows us for a simpler proof of the Soundness Theorem (Theorem 126). On the other side, the equivalence with the head reduction makes available the separation property (here Theorem 129) that Leventis proved for the head reduction strategy [60] and that will play a crucial role for completeness.

5.1.3 Context equivalence

A standard way of comparing terms is by observing their behaviours within contexts. Intuitively, two terms M and N are considered as equivalent if any occurrence of M in another term L can be replaced with N without changing the observable behaviour of L. The typical observation in $\Lambda_\oplus$ is the probability of converging to a value. Since in this setting values are head normal forms, *context preorder* $\leq_{\mathrm{cxt}}$ and *context equivalence* $=_{\mathrm{cxt}}$ can be defined as follows:

Definition 73 (Context equivalence)**.** For every $M, N \in \Lambda_\oplus$:

(1) *Context preorder*: $M \leq_{\mathrm{cxt}} N$ if and only if, for all $\mathcal{C} \in \mathsf{C}\Lambda_\oplus$, $\sum[\![\mathcal{C}[M]]\!] \leq \sum[\![\mathcal{C}[N]]\!]$;

(2) *Context equivalence*: $M =_{\mathrm{cxt}} N$ if and only if, for all $\mathcal{C} \in \mathsf{C}\Lambda_\oplus$, $\sum[\![\mathcal{C}[M]]\!] = \sum[\![\mathcal{C}[N]]\!]$.

Note that $M =_{\mathrm{cxt}} N$ if and only if $M \leq_{\mathrm{cxt}} N$ and $N \leq_{\mathrm{cxt}} M$.

Example 26. Consider the terms $M \triangleq \lambda xyz.z(x \oplus y)$ and $N \triangleq \lambda xyz.(zx \oplus zy)$. They can be discriminated by the context $\mathcal{C} \triangleq [\cdot]\mathbf{\Omega}\mathbf{I}\mathbf{\Delta}$, where $\mathbf{\Omega}$, $\mathbf{I}$, and $\mathbf{\Delta}$ are as in Example 22. In Figure 5.3 we show that $\sum[\![\mathcal{C}[M]]\!] = \frac{1}{4}$ and $\sum[\![\mathcal{C}[N]]\!] = \frac{1}{2}$.

$$\begin{array}{c}\begin{array}{c}\vdots\\ M \Downarrow M\end{array} \quad \begin{array}{c}\begin{array}{c}\vdots\\ \lambda yz.z(\boldsymbol{\Omega}\oplus y) \Downarrow \lambda yz.z(\boldsymbol{\Omega}\oplus y)\end{array} \quad \begin{array}{c}\begin{array}{c}\vdots\\ \lambda z.z\,\mathsf{hid} \Downarrow \lambda z.z\,\mathsf{hid}\end{array} \quad \begin{array}{c}\begin{array}{c}\vdots\\ \boldsymbol{\Delta} \Downarrow \boldsymbol{\Delta}\end{array} \quad \begin{array}{c}\begin{array}{c}\begin{array}{c} \\ \hline \boldsymbol{\Omega} \Downarrow \bot\end{array}\,s1 \quad \begin{array}{c}\vdots\\ \mathbf{I} \Downarrow \mathbf{I}\end{array} \\ \hline \mathsf{hid} \Downarrow \frac{1}{2}\cdot\mathbf{I}\end{array}\,s5 \quad \begin{array}{c}\vdots\\ \mathsf{hid} \Downarrow \frac{1}{2}\cdot\mathbf{I}\end{array} \\ \hline \mathsf{hid}\,\mathsf{hid} \Downarrow \frac{1}{4}\cdot\mathbf{I}\end{array}\,s4 \\ \hline \boldsymbol{\Delta}\,\mathsf{hid} \Downarrow \frac{1}{4}\cdot\mathbf{I}\end{array}\,s4 \\ \hline (\lambda z.z\,\mathsf{hid})\boldsymbol{\Delta} \Downarrow \frac{1}{4}\cdot\mathbf{I}\end{array}\,s4 \\ \hline (\lambda yz.z(\boldsymbol{\Omega}\oplus y))\mathbf{I}\boldsymbol{\Delta} \Downarrow \frac{1}{4}\cdot\mathbf{I}\end{array}\,s4 \\ \hline M\boldsymbol{\Omega}\mathbf{I}\boldsymbol{\Delta} \Downarrow \frac{1}{4}\cdot\mathbf{I}\end{array}\,s4$$

$$\begin{array}{c}\begin{array}{c}\begin{array}{c}\begin{array}{c}\begin{array}{c} \\ \hline z \Downarrow z\end{array}\,s2 \\ \hline zx \Downarrow zx\end{array}\,s4 \quad \begin{array}{c}\begin{array}{c} \\ \hline z \Downarrow z\end{array}\,s2 \\ \hline zy \Downarrow zy\end{array}\,s4 \\ \hline zx\oplus zy \Downarrow \frac{1}{2}\cdot zx + \frac{1}{2}\cdot zy\end{array}\,s5 \\ \hline\hline N \Downarrow \frac{1}{2}\cdot\lambda xyz.zx + \frac{1}{2}\cdot\lambda xyz.zy\end{array}\,s3 \quad \begin{array}{c} \\ \hline (\lambda yz.z\boldsymbol{\Omega})\mathbf{I}\Delta \Downarrow \bot\end{array}\,s1 \quad \begin{array}{c}\begin{array}{c}\vdots\\ \lambda yz.zy \Downarrow \lambda yz.zy\end{array} \quad \begin{array}{c}\begin{array}{c}\vdots\\ \lambda z.z\mathbf{I} \Downarrow \lambda z.z\mathbf{I}\end{array} \quad \begin{array}{c}\begin{array}{c}\vdots\\ \boldsymbol{\Delta} \Downarrow \boldsymbol{\Delta}\end{array} \quad \begin{array}{c}\begin{array}{c}\vdots\\ \mathbf{I} \Downarrow \mathbf{I}\end{array} \quad \begin{array}{c}\vdots\\ \mathbf{I} \Downarrow \mathbf{I}\end{array} \\ \hline \mathbf{I}\mathbf{I} \Downarrow \mathbf{I}\end{array}\,s4 \\ \hline \boldsymbol{\Delta}\mathbf{I} \Downarrow \mathbf{I}\end{array}\,s4 \\ \hline (\lambda z.z\mathbf{I})\boldsymbol{\Delta} \Downarrow \mathbf{I}\end{array}\,s4 \\ \hline (\lambda yz.zy)\mathbf{I}\boldsymbol{\Delta} \Downarrow \mathbf{I}\end{array}\,s4 \\ \hline N\boldsymbol{\Omega}\mathbf{I}\Delta \Downarrow \frac{1}{2}\cdot\mathbf{I}\end{array}\,s4$$

Figure 5.3: The derivations in the big-step semantics of $M\boldsymbol{\Omega}\mathbf{I}\boldsymbol{\Delta} \Downarrow \frac{1}{4}\cdot\mathbf{I}$ and $N\boldsymbol{\Omega}\mathbf{I}\boldsymbol{\Delta} \Downarrow \frac{1}{2}\cdot\mathbf{I}$, where $M \triangleq \lambda xyz.z(x\oplus y)$, $N \triangleq \lambda xyz.(zx\oplus zy)$, $\boldsymbol{\Delta} = \lambda x.xx$, and $\mathsf{hid} = \boldsymbol{\Omega}\oplus\mathbf{I}$. The double inference line means multiple applications of the same rule.

Contexts enjoy the following monotonicity property:

Lemma 99. *Let $M, N \in \Lambda_{\oplus}$:*

(1) if $[\![M]\!] \leq_{\mathfrak{D}} [\![N]\!]$ then $\forall \mathcal{C} \in \mathsf{C}\Lambda_{\oplus}$, $[\![\mathcal{C}[M]]\!] \leq_{\mathfrak{D}} [\![\mathcal{C}[N]]\!]$;

(2) if $[\![M]\!] = [\![N]\!]$ then $\forall \mathcal{C} \in \mathsf{C}\Lambda_{\oplus}$, $[\![\mathcal{C}[M]]\!] = [\![\mathcal{C}[N]]\!]$.

Proof. Point (2) follows from point (1). Concerning the latter, we prove it by structural induction on the context $\mathcal{C} \in \mathsf{C}\Lambda_{\oplus}$. The case $\mathcal{C} = [\cdot]$ is trivial. Let $\mathcal{C} = \lambda x.\mathcal{C}'$ and let $\mathscr{D}$ be such that $\lambda x.\mathcal{C}'[M] \Downarrow \mathscr{D}$. By Proposition 97.(3) there exists $\mathscr{D}'$ such that $\mathcal{C}'[M] \Downarrow \mathscr{D}'$ and $\mathscr{D} \leq_{\mathfrak{D}} \lambda x.\mathscr{D}'$. By induction hypothesis, there exists $\mathscr{E}'$ such that $\mathcal{C}'[N] \Downarrow \mathscr{E}'$ and $\mathscr{D}' \leq_{\mathfrak{D}} \mathscr{E}'$. We define $\mathscr{E} \triangleq \lambda x.\mathscr{E}'$, so that $\lambda x.\mathcal{C}'[N] \Downarrow \mathscr{E}$ and $\mathscr{D} \leq_{\mathfrak{D}} \lambda x.\mathscr{D}' \leq_{\mathfrak{D}} \lambda x.\mathscr{E}' = \mathscr{E}$. We now consider the case $\mathcal{C} = \mathcal{C}'L$ (the case $\mathcal{C} = L\mathcal{C}'$ is similar). Let $\mathscr{D}$ be such that $\mathcal{C}'[M]L \Downarrow \mathscr{D}$. By Proposition 97.(1), there exist head distributions $\mathscr{D}'$ and $\{\mathscr{D}_{H,L}\}_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{D}')}$ such that $\mathcal{C}'[M] \Downarrow \mathscr{D}'$, $\{H[L/x] \Downarrow \mathscr{D}_{H,L}\}_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{D}')}$, and:

$$\mathscr{D} \leq_{\mathfrak{D}} \sum_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{D}')} \mathscr{D}'(\lambda x.H) \cdot \mathscr{D}_{H,L} + \sum_{H \,\in\, \mathrm{supp}(\mathscr{D}') \,\cap\, \mathrm{NEUT}} \mathscr{D}'(H) \cdot HL$$

By induction hypothesis, there exists a head distribution $\mathscr{E}'$ such that $\mathcal{C}'[N] \Downarrow \mathscr{E}'$ and $\mathscr{D}' \leq_{\mathfrak{D}} \mathscr{E}'$. For all $\lambda x.H \in \mathrm{supp}(\mathscr{E}')$, we set:

$$\mathscr{E}_{H,L} \triangleq \begin{cases} \mathscr{D}_{H,L} & \text{if } \lambda x.H \in \mathrm{supp}(\mathscr{D}'), \\ \perp & \text{otherwise.} \end{cases}$$

$$\mathscr{E} \triangleq \sum_{\lambda x.H \,\in\, \mathrm{supp}(\mathscr{E}')} \mathscr{E}'(\lambda x.H) \cdot \mathscr{E}_{H,L} + \sum_{H \,\in\, \mathrm{supp}(\mathscr{E}') \,\cap\, \mathrm{NEUT}} \mathscr{E}'(H) \cdot HL$$

Therefore, $\mathcal{C}'[N]L \Downarrow \mathscr{E}$ and $\mathscr{D} \leq_{\mathfrak{D}} \mathscr{E}$. Finally, let us consider the case $\mathcal{C} = \mathcal{C}' \oplus L$ (the case $\mathcal{C} = L \oplus \mathcal{C}'$ is symmetric). Let $\mathscr{D}$ be such that $\mathcal{C}'[M] \oplus L \Downarrow \mathscr{D}$. By Proposition 97.(4), there exist $\mathscr{D}'$ and $\mathscr{D}''$ such that $\mathcal{C}'[M] \Downarrow \mathscr{D}'$, $L \Downarrow \mathscr{D}''$ and $\mathscr{D} \leq_{\mathfrak{D}} \frac{1}{2} \cdot \mathscr{D}' + \frac{1}{2} \cdot \mathscr{D}''$. By induction hypothesis, there exists $\mathscr{E}'$ such that $\mathcal{C}'[N] \Downarrow \mathscr{E}'$ and $\mathscr{D}' \leq_{\mathfrak{D}} \mathscr{E}'$. We define $\mathscr{E} = \frac{1}{2} \cdot \mathscr{E}' + \frac{1}{2} \cdot \mathscr{D}''$, so that $\mathcal{C}'[N] \oplus L \Downarrow \mathscr{E}$ and $\mathscr{D} \leq_{\mathfrak{D}} \frac{1}{2} \cdot \mathscr{D}' + \frac{1}{2} \cdot \mathscr{D}'' \leq_{\mathfrak{D}} \frac{1}{2} \cdot \mathscr{E}' + \frac{1}{2} \cdot \mathscr{D}'' = \mathscr{E}$. □

An immediate consequence of Lemma 99 is the soundness of the operational semantics:

Proposition 100. *Let $M, N \in \Lambda_{\oplus}$:*

(1) if $[\![M]\!] \leq_{\mathfrak{D}} [\![N]\!]$ then $M \leq_{\mathrm{cxt}} N$,

(2) if $[\![M]\!] = [\![N]\!]$ then $M =_{\mathrm{cxt}} N$.

Thanks to Proposition 100, one can prove that quite different terms are indeed context equivalent, as the following example shows:

Example 27. The term MM in Example 25 and y are context equivalent, i.e. $MM =_{\mathrm{cxt}} y$, since $[\![MM]\!] = y$.

However, not all context equivalent terms have the same semantics:

Example 28. The term $\lambda x.x$ and its η-expansion $\lambda xy.xy$ are context equivalent but $[\![\lambda x.x]\!] = \lambda x.x \neq \lambda xy.xy = [\![\lambda xy.xy]\!]$.

Proving context equivalence might be rather difficult since its definition quantifies over the set of *all* contexts. Fortunately, various other tools can be deployed to show the equivalence of terms. An example is bisimilarity, we shall discuss in the next subsection. Checking that two terms are bisimilar requires the *existence* of a particular relation, called "bisimulation". Proving that bisimilarity and context equivalence actually coincide would imply that the latter can be established using the much more tractable operational techniques coming from bisimilarity.

5.1.4 Probabilistic (bi)similarity

Following [27], we recall here the main definitions and basic properties about labelled Markov chains and its associated probabilistic (bi)similarity [58], as these do not depend on a specific operational semantics. In the next subsection, we shall apply these notions to the operational semantics of $\Lambda_\oplus$, getting the probabilistic applicative (bi)similarity.

In Section 2.5.2 we introduced the basic definitions and conventions concerning relations. So, for example, $\mathcal{R}(X)$ stands for the image of X under $\mathcal{R}$, $\mathcal{R}^{op}$ is the converse of $\mathcal{R}$, and $X/\mathcal{R}$ denotes the set of all equivalence classes modulo $\mathcal{R}$, provided that the latter is an equivalence relation.

Definition 74 (Labelled Markov chain)**.** A *labelled Markov chain* is a triple $\mathcal{M} = (\mathcal{S}, \mathcal{L}, \mathcal{P})$, where $\mathcal{S}$ is a countable set of states, $\mathcal{L}$ is a set of labels (actions) and $\mathcal{P}$ is a transition probability matrix, i.e. a function $\mathcal{P} : \mathcal{S} \times \mathcal{L} \times \mathcal{S} \longrightarrow [0, 1]$ satisfying the following condition:

$$\forall s \in \mathcal{S},\ \forall l \in \mathcal{L} : \qquad \sum_{t \in \mathcal{S}} \mathcal{P}(s, l, t) \leq 1.$$

We let the expression $\mathcal{P}(s, l, X)$ denote $\sum_{t \in X} \mathcal{P}(s, l, t)$.

Probabilistic simulation and probabilistic bisimulation can be defined as follows:

Definition 75 (Probabilistic (bi)simulation)**.** Let $(\mathcal{S}, \mathcal{L}, \mathcal{P})$ be a labelled Markov chain and $\mathcal{R}$ be a relation over $\mathcal{S}$:

(1) $\mathcal{R}$ is a *probabilistic simulation* if it is a preorder and it satisfies the following condition:

$$\forall (s, t) \in \mathcal{R},\ \forall X \subseteq \mathcal{S},\ \forall l \in \mathcal{L} : \qquad \mathcal{P}(s, l, X) \leq \mathcal{P}(t, l, \mathcal{R}(X)).$$

(2) $\mathcal{R}$ is a *probabilistic bisimulation* if it is an equivalence and it satisfies the following condition:

$$\forall (s, t) \in \mathcal{R},\ \forall E \in \mathcal{S}/\mathcal{R},\ \forall l \in \mathcal{L} : \qquad \mathcal{P}(s, l, E) = \mathcal{P}(t, l, E).$$

Probabilistic (bi)similarity is the union of all probabilistic (bi)simulations.

Definition 76 (Probabilistic (bi)similarity)**.** Let $(\mathcal{S}, \mathcal{L}, \mathcal{P})$ be a labelled Markov chain. Then, for all $s, t \in \mathcal{S}$:

(1) *Probabilistic similarity*: $s \precsim t$ if and only if $\exists \mathcal{R}$ probabilistic simulation such that $s\ \mathcal{R}\ t$;

(2) *Probabilistic bisimilarity*: $s \sim t$ if and only if $\exists \mathcal{R}$ probabilistic bisimulation such that $s\ \mathcal{R}\ t$.

We shall prove that both $\precsim$ and $\sim$ are in turn, respectively, a probabilistic simulation and a probabilistic bisimulation. Now, a probabilistic bisimulation has to be, by definition, an *equivalence relation*, and the union of two equivalence relations is not in general an equivalence relation. The following is a standard way to overcome the problem:

Lemma 101. *If $\{\mathcal{R}_i\}_{i \in I}$ is a collection of probabilistic bisimulations, then also their reflexive and transitive closure $(\bigcup_{i \in I} \mathcal{R}_i)^*$ is a probabilistic bisimulation.*

Proof. Let us fix $\mathcal{T} \triangleq (\bigcup_{i \in I} \mathcal{R}_i)^*$, which is by definition reflexive and transitive. Let us prove that $\mathcal{T}$ is symmetric. If $(s, t) \in \mathcal{T}$ then there are $n \geq 0$ states $v_0, \dots, v_n$ such that $v_0 = s$, $v_n = t$ and, for all $1 \leq i \leq n$, there exists $j \in I$ such that $(v_{i-1,} v_i) \in \mathcal{R}_j$. By the symmetry of each of the $\mathcal{R}_j$, we easily get that $(v_i, v_{i-1}) \in \mathcal{R}_j$. As a consequence, we have $(t, s) \in \mathcal{T}$. Now, let $(s, t) \in \mathcal{T}$, $l \in \mathcal{L}$, and $E \in \mathcal{S}/\mathcal{T}$. Then, there are $n \geq 0$ states $v_0, \dots, v_n$ such that $v_0 = s$, $v_n = t$ and, for all $1 \leq i \leq n$, there exists $j \in I$ such that $(v_{i-1,} v_i) \in \mathcal{R}_j$. We have $\mathcal{P}(s, l, E) = \mathcal{P}(v_0, l, E) = \ldots = \mathcal{P}(v_n, l, E) = \mathcal{P}(t, l, E)$. □

Proposition 102. *The relation $\sim$ is a probabilistic bisimulation.*

Proof. By Lemma 101, $(\sim)^*$ is a probabilistic bisimulation, so that it suffices to prove that $\sim = (\sim)^*$. On the one hand, we clearly have $\sim \subseteq (\sim)^*$. On the other hand, since $(\sim)^*$ is a probabilistic bisimulation, it is included in the union of them all, that is, $\sim$. □

The following lemma is analogous to Lemma 101.

Lemma 103. *If $\{\mathcal{R}_i\}_{i\in I}$ is a collection of probabilistic simulations, then also their reflexive and transitive closure $(\bigcup_{i\in I}\mathcal{R}_i)^*$ is a probabilistic simulation.*

Proof. $\mathcal{R} \triangleq (\bigcup_{i\in I}\mathcal{R}_i)^*$ is a preorder by construction. So, let $(s,t)\in\mathcal{R}$, and let $l\in\mathcal{L}$ and $X\subseteq\mathcal{S}$. Then there are $n\geq 0$ states $v_0,\dots,v_n$ such that $v_0 = s$, $v_n = t$ and, for all $1\leq i\leq n$, there exists $j_i\in I$ such that $v_{i-1}\,\mathcal{R}_{j_i}\,v_i$. As a consequence, for every $l\in\mathcal{L}$ and for every $X\subseteq\mathcal{S}$, we have:

$$\begin{aligned}\mathcal{P}(v_0,l,X) &\leq \mathcal{P}(v_1,l,\mathcal{R}_{j_1}(X)) \leq \mathcal{P}(v_2,l,\mathcal{R}_{j_2}(\mathcal{R}_{j_1}(X))) \leq \dots\\ &\dots \leq \mathcal{P}(v_n,l,\mathcal{R}_{j_n}(\dots(\mathcal{R}_{j_2}(\mathcal{R}_{j_1}(X)))))\end{aligned}$$

Since by definition $\mathcal{R}_{j_n}(\dots(\mathcal{R}_{j_2}(\mathcal{R}_{j_1}(X))))\subseteq\mathcal{R}(X)$, we have $\mathcal{P}(s,l,X)\leq\mathcal{P}(t,l,\mathcal{R}(X))$. □

Proposition 104. *The relation $\precsim$ is a probabilistic simulation.*

Proof. Similar to Proposition 102. □

We now prove that $\sim\, = \,\precsim\cap\precsim^{op}$. To begin with, we prove some preliminary lemmas.

Lemma 105. *If $\mathcal{R}$ is a symmetric probabilistic simulation, then $\mathcal{R}$ is a probabilistic bisimulation.*

Proof. If $\mathcal{R}$ is a symmetric probabilistic simulation, by definition, it is an equivalence relation. Now, let $(s,t)\in\mathcal{R}$, $l\in\mathcal{L}$, and $E\in\mathcal{S}/\mathcal{R}$. On the one hand, since $\mathcal{R}$ is a probabilistic simulation, we have $\mathcal{P}(s,l,E)\leq\mathcal{P}(t,l,\mathcal{R}(E))$, but $\mathcal{R}(E) = E$. Since $\mathcal{R}$ is symmetric, we also have $(t,s)\in\mathcal{R}$, which implies $\mathcal{P}(t,l,E)\leq\mathcal{P}(s,l,E)$. □

Lemma 106. *If $\mathcal{R}$ is a probabilistic bisimulation, then $\mathcal{R}$ and $\mathcal{R}^{op}$ are probabilistic simulations.*

Proof. Notice that, since $\mathcal{R}$ is symmetric by assumption, if $\mathcal{R}$ is a probabilistic simulation then $\mathcal{R}^{op}$ is a probabilistic simulation. So, let us prove that $\mathcal{R}$ is a probabilistic simulation. Clearly, it is a preorder. Let $(s,t)\in\mathcal{R}$, $l\in\mathcal{L}$, and $X\subseteq\mathcal{S}$. Consider the family $\{X_i\}_{i\in I}$ of all equivalence subclasses modulo $\mathcal{R}$ contained in X, i.e. for all $i\in I$, we have $X_i\subseteq E_i\in\mathcal{S}/\mathcal{R}$ and $X = \biguplus_{i\in I}X_i$. As a consequence, $\mathcal{R}(X) = \biguplus_{i\in I}E_i$. Therefore, we have:

$$\mathcal{P}(s,l,X) = \sum_{i\in I}\mathcal{P}(s,l,X_i) \leq \sum_{i\in I}\mathcal{P}(s,l,E_i) = \sum_{i\in I}\mathcal{P}(t,l,E_i) = \mathcal{P}(t,l,\mathcal{R}(X)).$$ □

Proposition 107. *It holds that $\sim\, = \,\precsim\cap\precsim^{op}$.*

Proof. The fact that $\sim$ is a subset of $\precsim\cap\precsim^{op}$ is a straightforward consequence of Lemma 106. Let us now prove the converse, i.e. that $\precsim\cap\precsim^{op}$ is a probabilistic bisimulation. Clearly, it is an equivalence relation. Now, let $(s,t)\in(\precsim\cap\precsim^{op})$, $l\in\mathcal{L}$, and $E\in\mathcal{S}/(\precsim\cap\precsim^{op})$. Define the following two sets of states $X\triangleq\,\precsim\!(E)$ and $Y\triangleq X - E$. Observe that Y and E are disjoint sets of states whose union is precisely X. Moreover, notice that both X and Y are closed w.r.t. $\precsim$:

- On the one hand, if $s\in\,\precsim\!(X)$, then $s\in\,\precsim\!(\precsim\!(E)) = \,\precsim\!(E) = X$;

- On the other hand, if $s \in \precsim(Y) = \precsim(X - E)$, then there is $t \in X$ which is not in E such that $t \precsim s$. But then s is itself in X by the previous point. Moreover, s cannot be in E because, otherwise, from $t \in X = \precsim(E)$ we would have $s \precsim t$, meaning that s and t are in the same equivalence class modulo $\precsim \cap \precsim^{op}$, so that $t \in E$. A contradiction.

As a consequence, we have:

$$\mathcal{P}(s, l, X) \leq \mathcal{P}(t, l, \precsim(X)) = \mathcal{P}(t, l, X)$$
$$\mathcal{P}(t, l, X) \leq \mathcal{P}(s, l, \precsim(X)) = \mathcal{P}(s, l, X)$$

Hence, $\mathcal{P}(s, l, X) = \mathcal{P}(t, l, X)$ and, similarly, $\mathcal{P}(s, l, Y) = \mathcal{P}(t, l, Y)$. Therefore, $\mathcal{P}(s, l, E) = \mathcal{P}(s, l, X) - \mathcal{P}(s, l, Y) = \mathcal{P}(t, l, X) - \mathcal{P}(t, l, Y) = \mathcal{P}(t, l, E)$. □

5.1.5 Probabilistic applicative (bi)similarity

In order to apply probabilistic (bi)similarity to $\Lambda_\oplus$, we need to present its operational semantics as a labelled Markov chain (Definition 77). Intuitively, terms are seen as states, while labels are of two kinds: one can either *evaluate* a term (this kind of transition will be labelled by τ), obtaining a distribution of head normal forms, or *apply* a head normal form to a term M (this kind of transition will be labelled by M).

This idea has been first developed in the standard λ-calculus by Abramsky [1], who called the corresponding notion of bisimilarity "applicative". Applicative bisimilarity has been then studied in the probabilistic λ-calculus for several reduction strategies like, for example, lazy call-by-name (Dal Lago et al. [27]) and call-by-value (Crubillé and Dal Lago [22]). The benefit of this approach is to check program equivalence via an *existential* quantifier (Definition 76.(2)) rather than a *universal* one, as in the case of context equivalence (Definition 73.(2)).

For technical reasons, it is useful to consider only closed terms and to consider for each closed head normal form $H = \lambda x.H'$ two distinct representations, depending on the way we consider it: either as a term or properly as a normal form, and in the latter case we indicate it as $\widetilde{H} \triangleq \nu x.H'$ to stress the difference. Consequently, we define $\widetilde{\mathrm{HNF}}$ as the set of all "distinguished" closed head normal form, namely $\{\widetilde{H} \mid H \in \mathrm{HNF}^\emptyset\}$. More in general, if $X \subseteq \mathrm{HNF}^\emptyset$, we define $\widetilde{X} \triangleq \{\widetilde{H} \mid H \in X\}$.

Definition 77 ($\Lambda_\oplus$-Markov chain)**.** The $\Lambda_\oplus$-*Markov chain* is the triple $(\Lambda_\oplus^\emptyset \uplus \widetilde{\mathrm{HNF}}, \Lambda_\oplus^\emptyset \uplus \{\tau\}, \mathcal{P}_\oplus)$, where the set of states is the disjoint union of the set of closed terms and the set of "distinguished" closed head normal forms, labels (actions) are either closed terms or the τ action, and the transition probability matrix $\mathcal{P}_\oplus$ is defined in the following way:

(i) for every closed term M and distinguished head normal form $\nu x.H$:

$$\mathcal{P}_\oplus(M, \tau, \nu x.H) \triangleq [\![M]\!](\lambda x.H),$$

(ii) for every closed term M and distinguished head normal form $\nu x.H$:

$$\mathcal{P}_\oplus(\nu x.H, M, H[M/x]) \triangleq 1,$$

(iii) in all other cases, $\mathcal{P}_\oplus$ returns 0.

Remark 16. In the $\Lambda_\oplus$-Markov chain, a term M can be thought of as at the head of a (potentially infinite) stack of applications, where at each time we first evaluate the head of the stack until we reach a head normal form H (point (i)), and then we apply H to the next term of the stack (point (ii)). This is exactly the behaviour of the head *spine* reduction on an application $MN_1 \dots N_n$. Lemma 125 formalizes these intuitions.

Since $\Lambda_\oplus$ can be seen as a labelled Markov chain, simulation and bisimulation can be defined as well:

Definition 78 (PAS and PAB)**.** A *probabilistic applicative (bi)simulation* is a probabilistic (bi)simulation of the $\Lambda_\oplus$-Markov chain. The *probabilistic applicative similarity*, PAS for short, and the *probabilistic applicative bisimilarity*, PAB for short, are defined as in Definition 76.(1) and Definition 76.(2).

From now on, with $\precsim$ (resp. $\sim$) we mean probabilistic *applicative* similarity (resp. bisimilarity).

The notions of PAS and PAB are defined on closed terms. We extend them to open terms in the following way:

Definition 79 (PAS and PAB for open terms)**.** Let $M, N \in \Lambda_\oplus^{\{x_1,\ldots,x_n\}}$. Then:

(1) $M \precsim N$ if and only if $\lambda x_1 \ldots x_n.M \precsim \lambda x_1 \ldots x_n.N$.

(2) $M \sim N$ if and only if $\lambda x_1 \ldots x_n.M \sim \lambda x_1 \ldots x_n.N$.

One can notice that the order of the abstractions in the term closure does not affect the obtained relation.

The following proposition is analogous to Proposition 100, stating the soundness of the operational semantics with respect to both PAS and PAB.

Proposition 108. *Let $M, N \in \Lambda_\oplus$:*

(1) if $[\![M]\!] \leq_{\mathfrak{D}} [\![N]\!]$ then $M \precsim N$,

(2) if $[\![M]\!] = [\![N]\!]$ then $M \sim N$.

Proof. We prove only the inequality soundness, as the equality one is an immediate consequence by Proposition 107. Let us first show point (1) for closed terms. So, suppose $M, N \in \Lambda_\oplus^\emptyset$ be such that $[\![M]\!] \leq_{\mathfrak{D}} [\![N]\!]$, and consider the relation $\mathcal{R} = \{(P,Q) \in \Lambda_\oplus^\emptyset \times \Lambda_\oplus^\emptyset \mid [\![P]\!] \leq_{\mathfrak{D}} [\![Q]\!]\} \cup \{(\nu x.H, \nu x.H) \in \widetilde{\mathrm{HNF}} \times \widetilde{\mathrm{HNF}}\}$. If we show that $\mathcal{R}$ is a PAS, then $\mathcal{R} \subseteq \precsim$, and hence $M \precsim N$. Clearly, $\mathcal{R}$ is a preorder. Now, let $(P,Q), (\nu x.H, \nu x.H) \in \mathcal{R}$, and let $X \subseteq \Lambda_\oplus^\emptyset \cup \widetilde{\mathrm{HNF}}$. It is straightforward that $\mathcal{P}_\oplus(\nu x.H, l, X) \leq \mathcal{P}_\oplus(\nu x.H, l, \mathcal{R}(X))$, for all $l \in \Lambda_\oplus^\emptyset \cup \{\tau\}$. Moreover, for all $F \in \Lambda_\oplus^\emptyset$ we have $0 = \mathcal{P}_\oplus(P, F, X) \leq \mathcal{P}_\oplus(Q, F, \mathcal{R}(X))$. Last:

$$\begin{aligned}\mathcal{P}_\oplus(P,\tau,X) = \sum_{\nu x.H \in X} \mathcal{P}_\oplus(P,\tau,\nu x.H) &= [\![P]\!](X \cap \mathrm{HNF}) \\ &\leq [\![Q]\!](X \cap \mathrm{HNF}) = \mathcal{P}_\oplus(Q,\tau,\mathcal{R}(X))\end{aligned}$$

Hence, for all $l \in \Lambda_\oplus^\emptyset \cup \{\tau\}$ and $X \subseteq \Lambda_\oplus^\emptyset \cup \widetilde{\mathrm{HNF}}$, we have $\mathcal{P}_\oplus(P,l,X) \leq \mathcal{P}_\oplus(Q,l,\mathcal{R}(X))$.
Now, let $M, N \in \Lambda_\oplus^{\{x_1,\ldots,x_n\}}$ be such that $[\![M]\!] \leq_{\mathfrak{D}} [\![N]\!]$. This means that $\lambda x_1 \ldots x_n.[\![M]\!] \leq_{\mathfrak{D}} \lambda x_1 \ldots x_n.[\![N]\!]$, and hence $[\![\lambda x_1 \ldots x_n.M]\!] \leq_{\mathfrak{D}} [\![\lambda x_1 \ldots x_n.N]\!]$ by Proposition 97.(3). Since these terms are closed, we have $\lambda x_1 \ldots x_n.M \precsim \lambda x_1 \ldots x_n.N$. By Definition 79, $M \precsim N$. □

Example 29. Let us show that $\mathbf{I} \sim \lambda xy.xy$ so that, from the soundness (Theorem 126), one can infer $\mathbf{I} =_{\mathrm{cxt}} \lambda xy.xy$. Let us define $\mathcal{R}_1 \triangleq \{(\mathbf{I}, \lambda xy.xy), (\lambda xy.xy, \mathbf{I})\}$, as well as $\mathcal{R}_2 \triangleq \{(\widetilde{\mathbf{I}}, \nu x.\lambda y.xy), (\nu x.\lambda y.xy, \widetilde{\mathbf{I}})\}$ and $\mathcal{R}_3 \triangleq \sim$. Let $\mathcal{R} \triangleq (\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3)^*$. Since $\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$ is a symmetric relation, then its reflexive and transitive closure $\mathcal{R} \triangleq (\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3)^*$ is an equivalence. Let us prove that it is a probabilistic bisimulation. We have to prove that $\mathcal{P}_\oplus(M, l, E) =$

$\mathcal{P}_\oplus(N, l, E)$, $\forall (M, N) \in \mathcal{R}$, $\forall E \in (\Lambda^\emptyset_\oplus \cup \widetilde{\mathrm{HNF}})/\mathcal{R}$, $\forall l \in \Lambda^\emptyset_\oplus \cup \{\tau\}$. Notice that, if this holds for $(M, N) \in (\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3)$, then we are done. Indeed, suppose $(M, N) \in \mathcal{R}$. Then there exists $n \geq 0$ and $P_0, \ldots, P_n \in \Lambda^\emptyset_\oplus \cup \widetilde{\mathrm{HNF}}$ such that $P_0 = M$, $P_n = N$ and $P_{i-1}\mathcal{R}_{j_i}P_i$ for every $1 \leq i \leq n$, where $1 \leq j_i \leq 3$. Hence, we have $\mathcal{P}_\oplus(M, l, E) = \mathcal{P}_\oplus(P_0, l, E) = \ldots = \mathcal{P}_\oplus(P_n, l, E) = \mathcal{P}_\oplus(N, l, E)$, $\forall E \in (\Lambda^\emptyset_\oplus \cup \widetilde{\mathrm{HNF}})/\mathcal{R}$, $\forall l \in \Lambda^\emptyset_\oplus \cup \{\tau\}$. Let us now show the case $(M, N) \in (\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3)$. If $(M, N) \in \mathcal{R}_3$ we just apply Proposition 102. Otherwise, it suffices to consider $(\mathbf{I}, \lambda xy.xy)$ and $(\widetilde{\mathbf{I}}, \nu x.\lambda y.xy)$. Recall that, by Definition 77, $\mathcal{P}_\oplus(M, N, E) = 0$ and $\mathcal{P}_\oplus(\widetilde{H}, \tau, E) = 0$, for all $M, N \in \Lambda^\emptyset_\oplus$, $\widetilde{H} \in \widetilde{\mathrm{HNF}}$ and $E \in (\Lambda^\emptyset_\oplus \cup \widetilde{\mathrm{HNF}})/\mathcal{R}$. On the one hand, since $(\widetilde{\mathbf{I}}, \nu x.\lambda y.xy) \in \mathcal{R}$, we have $\widetilde{\mathbf{I}} \in E$ if and only if $\nu x.\lambda y.xy \in E$, for all $E \in (\Lambda^\emptyset_\oplus \cup \widetilde{\mathrm{HNF}})/\mathcal{R}$. This implies $\mathcal{P}_\oplus(\mathbf{I}, \tau, E) = \mathcal{P}_\oplus(\lambda xy.xy, \tau, E)$, for all $E \in (\Lambda^\emptyset_\oplus \cup \widetilde{\mathrm{HNF}})/\mathcal{R}$. On the other hand, since terms are considered modulo renaming of bound variables, by Proposition 97 we have $[\![N]\!] = [\![\lambda y.Ny]\!]$, for all $N \in \Lambda^\emptyset_\oplus$ (notice that this equality may fail if N has free variables). By Proposition 108, $N \sim \lambda y.Ny$, and hence $N \in E$ if and only if $\lambda y.Ny \in E$, for all $E \in (\Lambda^\emptyset_\oplus \cup \widetilde{\mathrm{HNF}})/\mathcal{R}$. This implies $\mathcal{P}_\oplus(\widetilde{\mathbf{I}}, N, E) = \mathcal{P}_\oplus(\nu x.\lambda y.xy, N, E)$, for all $N \in \Lambda^\emptyset_\oplus$ and for all $E \in (\Lambda^\emptyset_\oplus \cup \widetilde{\mathrm{HNF}})/\mathcal{R}$.

Example 30. We show that the terms $M \triangleq \lambda xyz.z(x \oplus y)$ and $N \triangleq \lambda xyz.(zx \oplus zy)$ in Example 26 are not bisimilar. Indeed, suppose for the sake of contradiction that a probabilistic bisimulation $\mathcal{R}$ such that $(M, N) \in \mathcal{R}$ exists. By definition $\mathcal{R}$ is an equivalence relation. Let $E \in (\Lambda^\emptyset_\oplus \cup \widetilde{\mathrm{HNF}})/\mathcal{R}$ be such that $\nu x.\lambda yz.z(x \oplus y) \in E$. Then it must be that $\mathcal{P}_\oplus(M, \tau, E) = 1 = \mathcal{P}_\oplus(N, \tau, E)$, and it follows that both $\nu x.\lambda yz.zx$ and $\nu x.\lambda yz.zy$ are in E, so that $(\nu x.\lambda yz.z(x \oplus y), \nu x.\lambda yz.zx) \in \mathcal{R}$. Then it must be that $\mathcal{P}_\oplus(\nu x.\lambda yz.z(x \oplus y), \mathbf{\Omega}, E_1) = 1 = \mathcal{P}_\oplus(\nu x.\lambda yz.zx, \mathbf{\Omega}, E_1)$, for some $E_1 \in (\Lambda^\emptyset_\oplus \cup \widetilde{\mathrm{HNF}})/\mathcal{R}$ containing both $\lambda yz.z(\mathbf{\Omega} \oplus y)$ and $\lambda yz.z\mathbf{\Omega} \in E_1$, which implies $(\lambda yz.z(\mathbf{\Omega} \oplus y), \lambda yz.z\mathbf{\Omega}) \in \mathcal{R}$. By a similar reasoning, we get that $\mathcal{R}$ contains the pairs $(\nu y.\lambda z.z(\mathbf{\Omega} \oplus y), \nu y.\lambda z.z\mathbf{\Omega})$, $(\lambda z.z(\mathbf{\Omega} \oplus \mathbf{I}), \lambda z.z\mathbf{\Omega})$, and $(\nu z.z(\mathbf{\Omega} \oplus \mathbf{I}), \nu z.z\mathbf{\Omega})$. Now, let E_2 be an equivalence class containing $\mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I})$. From $\mathcal{P}_\oplus(\nu z.z(\mathbf{\Omega} \oplus \mathbf{I}), \mathbf{I}, E_2) = 1 = \mathcal{P}_\oplus(\nu z.z\mathbf{\Omega}, \mathbf{I}, E_2)$ we get that $\mathbf{I\Omega} \in E_2$, i.e. $(\mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I}), \mathbf{I\Omega}) \in \mathcal{R}$. Finally, if E_3 is an equivalence class such that $\nu x.x \in E_3$, then $\mathcal{P}_\oplus(\mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I}), \tau, E_3) = \frac{1}{2} = \mathcal{P}_\oplus(\mathbf{I\Omega}, \tau, E_3)$. This is a contradiction, since $\mathcal{P}_\oplus(\mathbf{I\Omega}, \tau, E_3) = 0$. Therefore, the terms M and N are not bisimilar.

5.2 The head spine reduction is equivalent to the head reduction

In the previous section we endowed the probabilistic λ-calculus with the big-step operational semantics $[\![\cdot]\!]$ introduced via the head spine reduction (Definition 71). This kind of semantics is often called "distribution-based" (see [19]), since it involves a relation between terms and distributions. The distribution-based semantics are opposed to the "term-based" ones (see [32]), which consider relations between terms weighted with probabilities. Examples of term-based semantics are the small-steps presentations of the head and head spine reduction strategies we gave in Definition 72.

In this section we show that the head and head spine reduction strategies yield the same operational semantics. First, we prove this property in a "term-based" setting, i.e. by considering the probabilistic transition relations in Definition 72. Actually, we shall establish an even stronger result: for all $n \in \mathbb{N}$ the probability that a term converges to a fixed head normal form in exactly n steps is the same for both strategies (Theorem 111). Then, we shall prove that the probabilistic transition relation corresponding to the head spine evaluation generates exactly the distribution-based semantics $[\![\cdot]\!]$ (Theorem 115).

5.2.1 Equivalence in a term-based setting

The following definition introduces the probability of convergence for both the head and head spine reduction strategies in a term-based setting.

Definition 80 ($\mathcal{H}^\infty$ and $\mathcal{S}^\infty$)**.** Let $M \in \Lambda_\oplus$, $H \in \mathrm{HNF}$ and $n \in \mathbb{N}$. We define the probability $\mathcal{H}^n(M,H)$ (resp. $\mathcal{S}^n(M,H)$) that M converges to H in exactly n steps of head reduction (resp. head spine reduction) as follows:

$$\mathcal{H}^n(M,H) \triangleq \sum_{\substack{(M_0,\dots,M_n) \text{ s.t. } M_0=M, \\ M_n=H, \forall i<n\, M_i \to_{p_{i+1}} M_{i+1}}} \prod_{i=1}^{n} p_i \qquad \mathcal{S}^n(M,H) \triangleq \sum_{\substack{(M_0,\dots,M_n) \text{ s.t. } M_0=M, \\ M_n=H, \forall i<n\, M_i \dashrightarrow_{p_{i+1}} M_{i+1}}} \prod_{i=1}^{n} p_i$$

The probability $\mathcal{H}^\infty(M,H)$ (resp. $\mathcal{S}^\infty(M,H)$) that M converges to H in an arbitrary number of steps of head reduction (resp. head spine reduction) is defined as follows:

$$\mathcal{H}^\infty(M,H) \triangleq \sum_{n=0}^{\infty} \mathcal{H}^n(M,H) \qquad\qquad \mathcal{S}^\infty(M,H) \triangleq \sum_{n=0}^{\infty} \mathcal{S}^n(M,H).$$

We now state and prove some basic properties about $\mathcal{H}^n$ and $\mathcal{S}^n$.

Lemma 109. *Let $M, N \in \Lambda_\oplus$ and $H \in \mathrm{HNF}$.*

(1) If either $\mathcal{X} = \mathcal{H}$ and $\mathcal{R} = \to$, or $\mathcal{X} = \mathcal{S}$ and $\mathcal{R} = \dashrightarrow$, then:

- *if $n = 0$ and $M = H$ then $\mathcal{X}^n(M,H) = 1$;*
- *if $n > 0$ and $M\ \mathcal{R}_1\ M'$ then $\mathcal{X}^n(M,H) = \mathcal{X}^{n-1}(M',H)$;*
- *if $n > 0$, $M\ \mathcal{R}_{\frac{1}{2}}\ M'$, $M\ \mathcal{R}_{\frac{1}{2}}\ M''$, then $\mathcal{X}^n(M,H) = \frac{1}{2}\cdot\mathcal{X}^{n-1}(M',H) + \frac{1}{2}\cdot\mathcal{X}^{n-1}(M'',H)$;*
- *in all other cases, $\mathcal{X}^n(M,H) = 0$.*

(2) For all $n \in \mathbb{N}$, $\mathcal{H}^n(\lambda x.M, \lambda x.H) = \mathcal{H}^n(M,H)$ and $\mathcal{S}^n(\lambda x.M, \lambda x.H) = \mathcal{S}^n(M,H)$.

(3) For all $n \in \mathbb{N}$, $\mathcal{H}^n(M[N/x],H) = \sum_{l+l'=n} \sum_{H' \in \mathrm{HNF}} \mathcal{H}^l(M,H') \cdot \mathcal{H}^{l'}(H'[N/x],H)$.

(4) For all $n \in \mathbb{N}$, $\mathcal{S}^n(MN,H) = \sum_{l+l'=n} \sum_{H' \in \mathrm{HNF}} \mathcal{S}^l(M,H') \cdot \mathcal{S}^{l'}(H'N,H)$.

Proof. Concerning point (1), we just prove the case where $n > 0$, $M \to_{\frac{1}{2}} M'$ and $M \to_{\frac{1}{2}} M''$:

$$\begin{aligned}
\mathcal{H}^n(M,H) &= \sum_{\substack{(M_0,\dots,M_n) \text{ s.t. } M_0=M, \\ M_n=H, \forall i<n\, M_i \to_{p_{i+1}} M_{i+1}}} \prod_{i=1}^{n} p_i \\
&= \frac{1}{2} \cdot \left(\sum_{\substack{(M_0,\dots,M_{n-1}) \text{ s.t. } M_0=M', \\ M_{n-1}=H, \forall i<n-1\, M_i \to_{p_{i+1}} M_{i+1}}} \prod_{i=1}^{n-1} p_i \right) \\
&\quad + \frac{1}{2} \cdot \left(\sum_{\substack{(M_0,\dots,M_{n-1}) \text{ s.t. } M_0=M'', \\ M_{n-1}=H, \forall i<n-1\, M_i \to_{p_{i+1}} M_{i+1}}} \prod_{i=1}^{n-1} p_i \right) \\
&= \frac{1}{2} \cdot \mathcal{H}^{n-1}(M',H) + \frac{1}{2} \cdot \mathcal{H}^{n-1}(M'',H).
\end{aligned}$$

Concerning point (2), for all $n \in \mathbb{N}$ we have:

$$\begin{aligned}
\mathcal{H}^n(M,H) &= \sum_{\substack{(M_0,\dots,M_n) \text{ s.t. } M_0=M,\\ M_n=H, \forall i<n\, M_i \to_{p_{i+1}} M_{i+1}}} \prod_{i=1}^{n} p_i \\
&= \sum_{\substack{(\lambda x.M_0,\dots,\lambda x.M_n) \text{ s.t. } \lambda x.M_0=\lambda x.M,\\ \lambda x.M_n=\lambda x.H, \forall i<n\, \lambda x.M_i \to_{p_{i+1}} \lambda x.M_{i+1}}} \prod_{i=1}^{n} p_i = \mathcal{H}^n(\lambda x.M, \lambda x.H).
\end{aligned}$$

We prove the equation $\mathcal{S}^n(M,H) = \mathcal{S}^n(\lambda x.M, \lambda x.H)$ in a similar way.

Let us now prove point (3) by induction on $n \in \mathbb{N}$. We have three cases:

- If M is a head normal form, then $\mathcal{H}^l(M,H') \neq 0$ just when $l = 0$ and $H' = M$. In all cases, the equation holds.
- Suppose $M \to_{\frac{1}{2}} M_1$ and $M \to_{\frac{1}{2}} M_2$. If $n = 0$ then the equation trivially holds. Otherwise, by Lemma 98.(2) we have $M[N/x] \to_{\frac{1}{2}} M_1[N/x]$ and $M[N/x] \to_{\frac{1}{2}} M_2[N/x]$. Therefore:

$$\begin{aligned}
\mathcal{H}^n(M[N/x],H) &= \frac{1}{2}\cdot\mathcal{H}^{n-1}(M_1[N/x],H) + \frac{1}{2}\cdot\mathcal{H}^{n-1}(M_2[N/x],H) && \text{point (1)}\\
&= \frac{1}{2}\cdot \sum_{l+l'=n-1}\sum_{H'\in \text{HNF}} \mathcal{H}^l(M_1,H')\cdot\mathcal{H}^{l'}(H'[N/x],H)\\
&\quad + \frac{1}{2}\cdot \sum_{l+l'=n-1}\sum_{H'\in \text{HNF}} \mathcal{H}^l(M_2,H')\cdot\mathcal{H}^{l'}(H'[N/x],H) && \text{IH}\\
&= \sum_{l+l'=n-1}\sum_{H'\in \text{HNF}} \mathcal{H}^{l+1}(M,H')\cdot\mathcal{H}^{l'}(H'[N/x],H) && \text{point (1)}\\
&= \sum_{l+l'=n}\sum_{H'\in \text{HNF}} \mathcal{H}^{l}(M,H')\cdot\mathcal{H}^{l'}(H'[N/x],H).
\end{aligned}$$

- If $M \to_1 M'$ then we proceed similarly.

Finally we prove point (4) by induction on $n \in \mathbb{N}$. We have three cases:

- If M is a head normal form, then $\mathcal{S}^n(M,H') \neq 0$ whenever $n = 0$ and $H' = M$. In all cases, the equation holds.
- Suppose $M \dashrightarrow_{\frac{1}{2}} M_1$ and $M \dashrightarrow_{\frac{1}{2}} M_2$. If $n = 0$ then the equation trivially holds. Otherwise, by Lemma 98.(1) we have $MN \dashrightarrow_{\frac{1}{2}} M_1N$ and $MN \dashrightarrow_{\frac{1}{2}} M_2N$. Therefore:

$$\begin{aligned}
\mathcal{S}^n(MN,H) &= \frac{1}{2}\cdot\mathcal{S}^{n-1}(M_1N,H) + \frac{1}{2}\cdot\mathcal{S}^{n-1}(M_2N,H) && \text{point (1)}\\
&= \frac{1}{2}\cdot \sum_{l+l'=n-1}\sum_{H'\in \text{HNF}} \mathcal{S}^l(M_1,H')\cdot\mathcal{S}^{l'}(H'N,H)\\
&\quad + \frac{1}{2}\cdot \sum_{l+l'=n-1}\sum_{H'\in \text{HNF}} \mathcal{S}^l(M_2,H')\cdot\mathcal{S}^{l'}(H'N,H) && \text{IH}\\
&= \sum_{l+l'=n-1}\sum_{H'\in \text{HNF}} \mathcal{S}^{l+1}(M,H')\cdot\mathcal{S}^{l'}(H'N,H) && \text{point (1)}\\
&= \sum_{l+l'=n}\sum_{H'\in \text{HNF}} \mathcal{S}^{l}(M,H')\cdot\mathcal{S}^{l'}(H'N,H).
\end{aligned}$$

- If $M \dashrightarrow_1 M'$, we proceed similarly. $\square$

For all $n \in \mathbb{N}$, we can construct the probabilistic transition relation $\to^n$ (resp. $\dashrightarrow^n$) from the probabilistic transition relation $\to$ (resp $\dashrightarrow$) of Definition 72 (see Section 2.5.2).

Lemma 110. *If $M \dashrightarrow_p M'$ then there exists $n_0 \in \mathbb{N}$ and $M_0 \in \Lambda_\oplus$ such that $M \to_p^{n_0+1} M_0$ and $M' \to_1^{n_0} M_0$. Diagrammatically:*

$$\begin{tikzcd}
M \arrow[r, dashed, "p"'] \arrow[dr, "n_0+1", "p"'] & M' \arrow[d, "n_0", "1"'] \\
 & M_0
\end{tikzcd}$$

Proof. By induction on the structure of M. M cannot be a head normal form, so that we have three cases:

- $M = \mathcal{E}[(\lambda y.H)Q]$, where $H \in \mathrm{HNF}$ and $\mathcal{E} = \lambda\vec{x}.[\cdot]\vec{L} \in \mathsf{H}\Lambda_\oplus$. Then, $M' = \mathcal{E}[H[Q/y]]$, and we set $n_0 \triangleq 0$ and $M_0 \triangleq M'$.

- $M = \mathcal{E}[(\lambda y.P)Q]$, where $P \dashrightarrow_p P'$ and $\mathcal{E} = \lambda\vec{x}.[\cdot]\vec{L} \in \mathsf{H}\Lambda_\oplus$. Then, $M' = \mathcal{E}[(\lambda y.P')Q]$. By repeatedly applying Lemma 98.(1), $P\vec{L} \dashrightarrow_p P'\vec{L}$. By induction hypothesis, there exist n_0' and P_0 such that $P\vec{L} \to_p^{n_0'+1} P_0$ and $P'\vec{L} \to_1^{n_0'} P_0$. By repeatedly applying Lemma 98.(2), we have that $P[Q/y]\vec{L} \to_p^{n_0'+1} P_0[Q/y]$ and $P'[Q/y]\vec{L} \to_1^{n_0'} P_0[Q/y]$, since y is not free in $\vec{L}$. Moreover, by repeatedly applying Lemma 98.(3), we have $\mathcal{E}[P[Q/y]] \to_p^{n_0'+1} \lambda\vec{x}.P_0[Q/y]$ and $\mathcal{E}[P'[Q/y]] \to_1^{n_0'} \lambda\vec{x}.P_0[Q/y]$. We set $n_0 \triangleq n_0' + 1$ and $M_0 \triangleq \lambda\vec{x}.P_0[Q/y]$. On the one hand, $\mathcal{E}[(\lambda y.P)Q] \to_1 \mathcal{E}[P[Q/y]] \to_p^{n_0'+1} \lambda\vec{x}.P_0[Q/y]$ and, on the other hand, $\mathcal{E}[(\lambda y.P')Q] \to_1 \mathcal{E}[P'[Q/y]] \to_1^{n_0'} \lambda\vec{x}.P_0[Q/y]$.

- $M = \mathcal{E}[P_1 \oplus P_2]$, where $\mathcal{E} = \lambda\vec{x}.[\cdot]\vec{L} \in \mathsf{H}\Lambda_\oplus$. Then, $M' = \mathcal{E}[P_i]$. We set $n_0 \triangleq 0$ and $M_0 \triangleq M'$. $\square$

The head and head spine reductions rewrite a given term to a given head normal form with the same probability. Even better, the probability of converging to this head normal form remains the same when reductions of a fixed length are considered.

Theorem 111 ($\mathcal{H}^n = \mathcal{S}^n$)**.** *Let $M \in \Lambda_\oplus$ and $H \in \mathrm{HNF}$. Then, for all $n \in \mathbb{N}$:*

$$\mathcal{S}^n(M, H) = \mathcal{H}^n(M, H).$$

Proof. By induction on n. If $n = 0$ then $\mathcal{S}^0(M,H) = \mathcal{H}^0(M,H)$ by definition. Suppose $n > 0$. If M is a head normal form, then $\mathcal{S}^n(M,H) = 0 = \mathcal{H}^n(M,H)$. Otherwise, we can apply a head spine reduction step to M. If $M \dashrightarrow_1 M'$ then, by Lemma 110, there exist n_0 and M_0 such that:

$$M \to_1^{n_0+1} M_0 \qquad M' \to_1^{n_0} M_0$$

Moreover, by induction hypothesis and by Lemma 109.(1) we have $\mathcal{S}^n(M,H) = \mathcal{S}^{n-1}(M',H) = \mathcal{H}^{n-1}(M',H)$. If $n_0 \leq n-1$ then $\mathcal{H}^{n-1}(M',H) = \mathcal{H}^{n-1-n_0}(M_0,H) = \mathcal{H}^n(M,H)$. Otherwise, $n - 1 < n_0$ and $\mathcal{H}^{n-1}(M,H) = 0 = \mathcal{H}^n(M,H)$.

If $M \dashrightarrow_{\frac{1}{2}} M'$ and $M \dashrightarrow_{\frac{1}{2}} M''$ then, by Lemma 110, there exist n'_0, n''_0 and M'_0, M''_0 such that:

$$M \rightarrow_{\frac{1}{2}}^{n'_0+1} M'_0 \quad M' \rightarrow_{1}^{n'_0} M'_0$$

$$M \rightarrow_{\frac{1}{2}}^{n''_0+1} M''_0 \quad M'' \rightarrow_{1}^{n''_0} M''_0$$

Then, there exist N, N' and N'' such that:

$$M \rightarrow_{1}^{t} N \qquad N \rightarrow_{\frac{1}{2}} N' \rightarrow_{1}^{t'} M'_0 \qquad N \rightarrow_{\frac{1}{2}} N'' \rightarrow_{1}^{t''} M''_0$$

where $n'_0 = t + t'$ and $n''_0 = t + t''$. By induction hypothesis and by Lemma 109.(1):

$$\begin{aligned}\mathcal{S}^n(M,H) &= \frac{1}{2} \cdot \mathcal{S}^{n-1}(M',H) + \frac{1}{2} \cdot \mathcal{S}^{n-1}(M'',H) \\ &= \frac{1}{2} \cdot \mathcal{H}^{n-1}(M',H) + \frac{1}{2} \cdot \mathcal{H}^{n-1}(M'',H)\end{aligned}$$

and we have four cases:

- If $n'_0, n''_0 \leq n-1$ then, by using Lemma 109.(1):

$$\begin{aligned}\mathcal{H}^n(M,H) &= \mathcal{H}^{n-t}(N,H) \\ &= \frac{1}{2} \cdot \mathcal{H}^{n-(t+1)}(N',H) + \frac{1}{2} \cdot \mathcal{H}^{n-(t+1)}(N'',H) \\ &= \frac{1}{2} \cdot \mathcal{H}^{n-(n'_0+1)}(M'_0,H) + \frac{1}{2} \cdot \mathcal{H}^{n-(n''_0+1)}(M''_0,H) \\ &= \frac{1}{2} \cdot \mathcal{H}^{n-1}(M',H) + \frac{1}{2} \cdot \mathcal{H}^{n-1}(M'',H).\end{aligned}$$

- If $n'_0 \leq n-1$ and $n-1 < n''_0$ then, by using Lemma 109.(1):

$$\begin{aligned}\mathcal{H}^n(M,H) &= \mathcal{H}^{n-t}(N,H) \\ &= \frac{1}{2} \cdot \mathcal{H}^{n-(t+1)}(N',H) + \frac{1}{2} \cdot \mathcal{H}^{n-(t+1)}(N'',H) \\ &= \frac{1}{2} \cdot \mathcal{H}^{n-(n'_0+1)}(M'_0,H) = \frac{1}{2} \cdot \mathcal{H}^{n-1}(M',H) \\ &= \frac{1}{2} \cdot \mathcal{H}^{n-1}(M',H) + \frac{1}{2} \cdot \mathcal{H}^{n-1}(M'',H).\end{aligned}$$

- The case where $n-1 < n'_0$ and $n''_0 \leq n-1$ is similar to the previous one.
- If $n-1 < n'_0, n''_0$ then $\mathcal{H}^n(M,H) = 0 = \frac{1}{2} \cdot \mathcal{H}^{n-1}(M',H) + \frac{1}{2} \cdot \mathcal{H}^{n-1}(M'',H)$. □

5.2.2 The term-based and the distribution-based semantics coincide

What we have established so far is an equivalence between the head and head spine reductions in a "term-based" operational semantics introduced through the notion of probabilistic transition relation. We are going to show that the term-based and the distribution-based semantics for the head spine reduction coincide. This allows us to show that the big-step semantics introduced in Definition 71 is invariant with respect to the usual head reduction step $(\lambda x.M)N \rightarrow M[N/x]$, where M is not necessarily a head normal form.

Lemma 112. *Let $M \in \Lambda_\oplus$. For all $H \in \mathrm{HNF}$, $[\![M]\!](H) \leq \mathcal{S}^\infty(M, H)$.*

Proof. We show that, for all $\mathscr{D}$ such that $M \Downarrow \mathscr{D}$ and for all $H \in \mathrm{HNF}$, it holds that $\mathscr{D}(H) \leq \mathcal{S}^\infty(M, H)$. The proof is by induction on the derivation of $M \Downarrow \mathscr{D}$ by considering the structure of M. Since the case $\mathscr{D} = \bot$ is trivial, we shall assume that the last rule of $M \Downarrow \mathscr{D}$ is not $s1$:

- If $M = x$ then $\mathscr{D} = x$ and the last rule of $M \Downarrow \mathscr{D}$ is $s2$. If $H \neq x$ then $\mathscr{D}(H) = 0$. Otherwise, $\mathscr{D}(x) = 1 = \mathcal{S}^\infty(x, x)$.
- If $M = \lambda x.M'$ then $\mathscr{D} = \lambda x.\mathscr{D}'$ and the last rule of $M \Downarrow \mathscr{D}$ is the following:

$$\frac{M' \Downarrow \mathscr{D}'}{\lambda x.M' \Downarrow \lambda x.\mathscr{D}'}\ s3$$

  If $H \in \mathrm{NEUT}$ then $\mathscr{D}(H) = 0$. Otherwise, $H = \lambda x.H'$ and, by using the induction hypothesis and Lemma 109.(2), we have:

$$\mathscr{D}(H) = (\lambda x.\mathscr{D}')(\lambda x.H') = \mathscr{D}'(H') \leq \mathcal{S}^\infty(M', H') = \mathcal{S}^\infty(\lambda x.M', \lambda x.H').$$

- If $M = PQ$ then the last rule of $M \Downarrow \mathscr{D}$ is as follows:

$$\frac{P \Downarrow \mathscr{E} \qquad \{H'[Q/x] \Downarrow \mathscr{F}_{H',Q}\}_{\lambda x.H' \in \mathrm{supp}(\mathscr{E})}}{PQ \Downarrow \sum_{\lambda x.H' \in \mathrm{supp}(\mathscr{E})} \mathscr{E}(\lambda x.H') \cdot \mathscr{F}_{H',Q} + \sum_{H' \in \mathrm{supp}(\mathscr{E}) \cap \mathrm{NEUT}} \mathscr{E}(H') \cdot H'Q}\ s4$$

  By using the induction hypothesis, Lemma 109.(1) and Lemma 109.(4), we have:

$$\begin{aligned} &\sum_{\lambda x.H' \in \mathrm{supp}(\mathscr{E})} \mathscr{E}(\lambda x.H') \cdot \mathscr{F}_{H',Q}(H) + \sum_{H' \in \mathrm{supp}(\mathscr{E}) \cap \mathrm{NEUT}} \mathscr{E}(H') \cdot H'Q(H) = \\ &\leq \sum_{\lambda x.H' \in \mathrm{HNF}} \mathcal{S}^\infty(P, \lambda x.H') \cdot \mathcal{S}^\infty(H'[Q/x], H) \ + \sum_{H' \in \mathrm{NEUT}} \mathcal{S}^\infty(P, H') \cdot \mathcal{S}^\infty(H'Q, H') \\ &\leq \sum_{\lambda x.H' \in \mathrm{HNF}} \mathcal{S}^\infty(P, \lambda x.H') \cdot \mathcal{S}^\infty((\lambda x.H')Q, H) \ + \sum_{H' \in \mathrm{NEUT}} \mathcal{S}^\infty(P, H') \cdot \mathcal{S}^\infty(H'Q, H') \\ &= \sum_{H' \in \mathrm{HNF}} \mathcal{S}^\infty(P, H') \cdot \mathcal{S}^\infty(H'Q, H) = \mathcal{S}^\infty(PQ, H) \end{aligned}$$

  and hence $\mathscr{D}(H) \leq \mathcal{S}^\infty(PQ, H)$.
- If $M = P \oplus Q$ then $\mathscr{D} = \frac{1}{2} \cdot \mathscr{D}_1 + \frac{1}{2} \cdot \mathscr{D}_2$ and the last rule of $M \Downarrow \mathscr{D}$ is as follows:

$$\frac{P \Downarrow \mathscr{D}_1 \qquad Q \Downarrow \mathscr{D}_2}{P \oplus Q \Downarrow \frac{1}{2} \cdot \mathscr{D}_1 + \frac{1}{2} \cdot \mathscr{D}_2}\ s5$$

  By using the induction hypothesis and by Lemma 109.(1), we have:

$$\mathscr{D}(H) = \frac{1}{2} \cdot \mathscr{D}_1(H) + \frac{1}{2} \cdot \mathscr{D}_2(H) \leq \frac{1}{2} \cdot \mathcal{S}^\infty(P, H) + \frac{1}{2} \cdot \mathcal{S}^\infty(Q, H) = \mathcal{S}^\infty(P \oplus Q, H).$$

□

Lemma 113. *Let $M \in \Lambda_\oplus$:*

(1) if $M \dashrightarrow_1 M'$ and $M' \Downarrow \mathscr{D}$, then $M \Downarrow \mathscr{D}$;

(2) if $M \dashrightarrow_{\frac{1}{2}} M_1$, $M \dashrightarrow_{\frac{1}{2}} M_2$, $M_1 \Downarrow \mathscr{D}_1$ and $M_2 \Downarrow \mathscr{D}_2$, then there exists $\mathscr{D}$ such that $\frac{1}{2} \cdot \mathscr{D}_1 + \frac{1}{2} \cdot \mathscr{D}_2 \leq \mathscr{D}$ and $M \Downarrow \mathscr{D}$.

Proof. We prove both points simultaneously by induction on the structure of M. If M is not a head normal form, then there exists a head context $\mathcal{E}$ such that $M = \mathcal{E}[P]$ and, either $P \dashrightarrow_1 P'$, or both $P \dashrightarrow_{\frac{1}{2}} P_1$ and $P \dashrightarrow_{\frac{1}{2}} P_2$. By looking at the structure of M we have several cases:

- If $\mathcal{E} = [\cdot]$ then we have three subcases:
  (a) If $M = (\lambda x.H)N$, then it must be that $M \dashrightarrow_1 M' = H[N/x]$. From $M' \Downarrow \mathscr{D}$ we can construct:

$$\dfrac{\dfrac{}{\lambda x.H \Downarrow \lambda x.H}\,s2 \qquad H[N/x] \Downarrow \mathscr{D}}{(\lambda x.H)N \Downarrow \mathscr{D}}\,s4$$

  (b) Suppose $M = (\lambda x.Q)N$ with $Q \notin \text{HNF}$. We consider the case where $Q \dashrightarrow_{\frac{1}{2}} Q_1$ and $Q \dashrightarrow_{\frac{1}{2}} Q_2$. W.l.o.g. we can assume that, for $i \in \{1,2\}$, the last rule of the derivation of $(\lambda x.Q_i)N \Downarrow \mathscr{D}_i$ is as follows:

$$\dfrac{\dfrac{Q_i \Downarrow \mathscr{E}_i}{\lambda x.Q_i \Downarrow \lambda x.\mathscr{E}_i}\,s3 \qquad \{H[N/x] \Downarrow \mathscr{F}^i_{H,N}\}_{\lambda x.H \,\in\, \text{supp}(\lambda x.\mathscr{E}_i)}}{(\lambda x.Q_i)N \Downarrow \sum_{\lambda x.H \,\in\, \text{supp}(\lambda x.\mathscr{E}_i)} (\lambda x.\mathscr{E}_i)(\lambda x.H) \cdot \mathscr{F}^i_{H,N}}\,s4$$

By applying the induction hypothesis, there exists $\mathscr{E}$ such that $Q \Downarrow \mathscr{E}$ and $\frac{1}{2} \cdot \mathscr{E}_1 + \frac{1}{2} \cdot \mathscr{E}_2 \leq \mathscr{E}$. By Lemma 96, for all $H \in \text{supp}(\mathscr{E}_1) \cap \text{supp}(\mathscr{E}_2)$ there exists $\mathscr{G}_{H,N}$ such that $H[N/x] \Downarrow \mathscr{G}_{H,N}$ and $\mathscr{F}^1_{H,N}, \mathscr{F}^2_{H,N} \leq \mathscr{G}_{H,N}$. We define:

$$\mathscr{F}_{H,N} \triangleq \begin{cases} \mathscr{F}^i_{H,N} & \text{if, for } i \in \{1,2\},\ H \in \text{supp}(\mathscr{E}_i) \text{ and } H \notin \text{supp}(\mathscr{E}_{3-i}), \\ \mathscr{G}_{H,N} & \text{if } H \in \text{supp}(\mathscr{E}_1) \cap \text{supp}(\mathscr{E}_2), \\ \perp & \text{otherwise.} \end{cases}$$

For all $H \in \text{supp}(\mathscr{E})$, we have $H[N/x] \Downarrow \mathscr{F}_{H,N}$. Moreover, for all $i \in \{1,2\}$ and $H \in \text{supp}(\mathscr{E}_i)$, $\mathscr{F}^i_{H,N} \leq \mathscr{F}_{H,N}$. We define $\mathscr{D} \triangleq \sum_{\lambda x.H \,\in\, \text{supp}(\lambda x.\mathscr{E})} (\lambda x.\mathscr{E})(\lambda x.H) \cdot \mathscr{F}_{H,N}$, so that $(\lambda x.Q)N \Downarrow \mathscr{D}$ and:

$$\begin{aligned}
\frac{1}{2} \cdot \mathscr{D}_1 + \frac{1}{2} \cdot \mathscr{D}_2 &= \frac{1}{2} \cdot \sum_{\lambda x.H \,\in\, \text{supp}(\lambda x.\mathscr{E}_1)} (\lambda x.\mathscr{E}_1)(\lambda x.H) \cdot \mathscr{F}^1_{H,N} \\
&\quad + \frac{1}{2} \cdot \sum_{\lambda x.H \,\in\, \text{supp}(\lambda x.\mathscr{E}_2)} (\lambda x.\mathscr{E}_2)(\lambda x.H) \cdot \mathscr{F}^2_{H,N} \\
&= \frac{1}{2} \cdot \sum_{H \,\in\, \text{supp}(\mathscr{E}_1)} \mathscr{E}_1(H) \cdot \mathscr{F}^1_{H,N} + \frac{1}{2} \cdot \sum_{H \,\in\, \text{supp}(\mathscr{E}_2)} \mathscr{E}_2(H) \cdot \mathscr{F}^2_{H,N} \\
&\leq \frac{1}{2} \cdot \sum_{H \,\in\, \text{supp}(\mathscr{E}_1)} \mathscr{E}_1(H) \cdot \mathscr{F}_{H,N} + \frac{1}{2} \cdot \sum_{H \,\in\, \text{supp}(\mathscr{E}_2)} \mathscr{E}_2(H) \cdot \mathscr{F}_{H,N} \\
&= \sum_{H \,\in\, \text{supp}(\mathscr{E})} \left(\frac{1}{2} \cdot \mathscr{E}_1 + \frac{1}{2} \cdot \mathscr{E}_2\right)(H) \cdot \mathscr{F}_{H,N}
\end{aligned}$$

$$\leq \sum_{H \in \mathrm{supp}(\mathscr{E})} \mathscr{E}(H) \cdot \mathscr{F}_{H,N} = \sum_{\lambda x.H \in \mathrm{supp}(\lambda x.\mathscr{E})} (\lambda x.\mathscr{E})(\lambda x.H) \cdot \mathscr{F}_{H,N}$$
$$= \mathscr{D}.$$

(c) If $M = P_1 \oplus P_2$ then it must be that $M \dashrightarrow_{\frac{1}{2}} M_1 = P_1$ and $M \dashrightarrow_{\frac{1}{2}} M_2 = P_2$, with $M_1 \Downarrow \mathscr{D}_1$ and $M_2 \Downarrow \mathscr{D}_2$. In this case, it suffices to define $\mathscr{D} \triangleq \frac{1}{2} \cdot \mathscr{D}_1 + \frac{1}{2} \cdot \mathscr{D}_2$.

- Suppose $\mathcal{E} = \lambda x.\mathcal{E}'$ and let us consider the case $P \dashrightarrow_{\frac{1}{2}} P_1$ and $P \dashrightarrow_{\frac{1}{2}} P_2$. Then, for $i \in \{1,2\}$, the last rule in the derivation of $\mathcal{E}[P_i] \Downarrow \mathscr{D}_i$ is as follows:

$$\frac{\mathcal{E}'[P_i] \Downarrow \mathscr{D}_i'}{\lambda x.\mathcal{E}'[P_i] \Downarrow \lambda x.\mathscr{D}_i'}\, s3$$

  By applying the induction hypothesis, there exists $\mathscr{D}'$ such that $\mathcal{E}'[P] \Downarrow \mathscr{D}'$ and $\frac{1}{2} \cdot \mathscr{D}_1' + \frac{1}{2} \cdot \mathscr{D}_2' \leq \mathscr{D}'$. We define $\mathscr{D} \triangleq \lambda x.\mathscr{D}'$. Then, we have both $\lambda x.\mathcal{E}'[P] \Downarrow \mathscr{D}$ and $\frac{1}{2} \cdot \mathscr{D}_1 + \frac{1}{2} \cdot \mathscr{D}_2 \leq \mathscr{D}$.

- Suppose $\mathcal{E} = \mathcal{E}'L$ and let us consider the case $P \dashrightarrow_{\frac{1}{2}} P_1$ and $P \dashrightarrow_{\frac{1}{2}} P_2$. Then, for $i \in \{1,2\}$, the last rule of the derivation of $\mathcal{E}[P_i] \Downarrow \mathscr{D}_i$ is as follows:

$$\frac{\mathcal{E}'[P_i] \Downarrow \mathscr{E}_i \qquad \{H'[L/x] \Downarrow \mathscr{F}^i_{H,L}\}_{\lambda x.H' \in \mathrm{supp}(\mathscr{E}_i)}}{\mathcal{E}'[P_i]L \Downarrow \sum_{\lambda x.H' \in \mathrm{supp}(\mathscr{E}_i)} \mathscr{E}_i(\lambda x.H') \cdot \mathscr{F}^i_{H',L} + \sum_{H' \in \mathrm{supp}(\mathscr{E}_i) \cap \mathrm{NEUT}} \mathscr{E}_i(H') \cdot H'L}\, s4$$

  The proof is similar to point (b). □

Lemma 114. *Let $M \in \Lambda_\oplus$. For all $H \in$ HNF, $\mathcal{S}^\infty(M,H) \leq [\![M]\!](H)$.*

Proof. We prove by induction on $n \in \mathbb{N}$ that there exists $\mathscr{D}$ such that $M \Downarrow \mathscr{D}$ and, $\forall H \in$ HNF, $\mathcal{S}^n(M,H) \leq \mathscr{D}(H)$. The case $n = 0$ is trivial, so let $n > 0$. If M is a head normal form, then $\mathcal{S}^n(M,H) = 0$ and we take $\mathscr{D} \triangleq \bot$. Otherwise, we have two cases:

- If $M \dashrightarrow_1 M'$ then $\mathcal{S}^n(M,H) = \mathcal{S}^{n-1}(M',H)$, by Lemma 109.(1). By induction hypothesis there exists $\mathscr{D}$ such that $M' \Downarrow \mathscr{D}$ and $\mathcal{S}^{n-1}(M',H) \leq \mathscr{D}(H)$, for all $H \in$ HNF. By applying Lemma 113.(1), $M \Downarrow \mathscr{D}$.

- If $M \dashrightarrow_{\frac{1}{2}} M'$ and $M \dashrightarrow_{\frac{1}{2}} M''$ then, by Lemma 109.(1), we have $\mathcal{S}^n(M,H) = \frac{1}{2} \cdot \mathcal{S}^{n-1}(M',H) + \frac{1}{2} \cdot \mathcal{S}^{n-1}(M'',H)$. By induction hypothesis there exist $\mathscr{D}'$ and $\mathscr{D}''$ such that $M' \Downarrow \mathscr{D}'$, $M'' \Downarrow \mathscr{D}''$, $\mathcal{S}^{n-1}(M',H) \leq \mathscr{D}'(H)$, and $\mathcal{S}^{n-1}(M'',H) \leq \mathscr{D}''(H)$, for all $H \in$ HNF. By applying Lemma 113.(2), there exists $\mathscr{D}$ such that $M \Downarrow \mathscr{D}$ and $\frac{1}{2} \cdot \mathscr{D}' + \frac{1}{2} \cdot \mathscr{D}'' \leq \mathscr{D}$. □

We are now able to prove that $\mathcal{H}^\infty$, $\mathcal{S}^\infty$ and $[\![\cdot]\!]$ are all equivalent operational semantics:

Theorem 115 (Equivalence)**.** *Let $M \in \Lambda_\oplus$. For all $H \in$ HNF, $\mathcal{H}^\infty(M,H) = \mathcal{S}^\infty(M,H) = [\![M]\!](H)$.*

Proof. Let $H \in$ HNF. By Theorem 111, we have $\mathcal{H}^\infty(M,H) = \mathcal{S}^\infty(M,H)$. By Lemma 112 and Lemma 114, we have $\mathcal{S}^\infty(M,H) = [\![M]\!](H)$. □

As expected, Proposition 97.(2) says that the operational semantics $[\![\cdot]\!]$ in Definition 71 is invariant under the head spine reduction step that rewrites $(\lambda x.H)N$ into $H[N/x]$, where $H \in$ HNF. A consequence of Theorem 115 is that $[\![\cdot]\!]$ is also invariant under the usual head reduction step rewriting $(\lambda x.M)N$ into $M[N/x]$, where M is not necessarily a head normal form:

Corollary 116. *Let $M, N \in \Lambda_\oplus$. Then $[\![(\lambda x.M)N]\!] = [\![M[N/x]]\!]$.*

Proof. From Lemma 109.(1), we have $\mathcal{H}^n((\lambda x.M)N, H) = \mathcal{H}^{n-1}(M[N/x], H)$, for all $n \in \mathbb{N}$ and $H \in$ HNF. This means that $\mathcal{H}^\infty((\lambda x.M)N, H) = \mathcal{H}^\infty(M[N/x], H)$. We conclude by Theorem 115. □

5.3 Soundess

A fundamental technique to establish the soundness of applicative (bi)similarity is the so-called *Howe's method* [52]. This method shows that applicative bisimilarity is a *congruence*, i.e. an equivalence relation that respects the structure of terms, which is the hard part in the soundness proof. This technique has been used in e.g. [27, 22] for, respectively, the lazy cbn and cbv semantics of $\Lambda_\oplus$. We consider here a different approach. Following the reasoning by Abramsky and Ong [2], we shall first prove that $\precsim$ is included in $\leq_{\text{app}}$ (Lemma 125), which requires a technical Key Lemma (Lemma 124) specific to the probabilistic framework, and then we conclude by applying a Context Lemma (Lemma 119).

The Context Lemma says that the computational behaviour of the context semantics is *functional.* This property has also been called *operational extensionality* in Bloom [16]. Milner [71] proved a similar result in the case of simply typed combinatory algebra. To the best of our knowledge, the Context Lemma lacks a corresponding formulation in the probabilistic λ-calculus $\Lambda_\oplus$, so we prove it in the following subsection.

5.3.1 The Context Lemma

Context equivalence captures the intuitive idea that two programs are indistinguishable in all possible programming contexts. As already stressed, though context preorder and equivalence are clearly important, it is hard to reason about them *directly.* The Context Lemma states that only the subset of applicative contexts "really matter".

Definition 81 (Applicative contexts)**.** An *applicative context* is a context $\mathcal{C} \in \mathsf{C}\Lambda_\oplus$ of the form $(\lambda x_1 \ldots x_n.[\cdot])P_1 \ldots P_m$, where $n, m \in \mathbb{N}$ and $P_1 \ldots P_m \in \Lambda_\oplus^\emptyset$. We denote by $\mathsf{A}\Lambda_\oplus$ the set of all applicative contexts. For every $M, N \in \Lambda_\oplus$:

(1) *Applicative context preorder*: $M \leq_{\text{app}} N$ if and only if $\sum[\![\mathcal{C}[M]]\!] \leq \sum[\![\mathcal{C}[N]]\!]$, for all $\mathcal{C} \in \mathsf{A}\Lambda_\oplus$;

(2) *Applicative context equivalence*: $M =_{\text{app}} N$ if and only if $\sum[\![\mathcal{C}[M]]\!] = \sum[\![\mathcal{C}[N]]\!]$, for all $\mathcal{C} \in \mathsf{A}\Lambda_\oplus$.

Notice that $M =_{\text{app}} N$ if and only if $M \leq_{\text{app}} N$ and $N \leq_{\text{app}} M$.

Lemma 117. *Let $M, N \in \Lambda_\oplus^{\Gamma \cup \{x\}}$:*

(1) if $M \leq_{\text{app}} N$ then $\lambda x.M \leq_{\text{app}} \lambda x.N$;

(2) if $\lambda x.M \leq_{\text{cxt}} \lambda x.N$ then $M \leq_{\text{cxt}} N$;

(3) if $M \leq_{\text{cxt}} N$ then, for all $L \in \Lambda_\oplus$, $ML \leq_{\text{cxt}} NL$.

Proof. Concerning point (1), let us suppose that $\lambda x.M \leq_{\text{app}} \lambda x.N$ does not hold. Then, there exists an applicative context $\mathcal{C} = (\lambda x_1 \ldots x_n.[\cdot])P_1 \ldots P_m$ such that $\sum[\![\mathcal{C}[\lambda x.N]]\!] < \sum[\![\mathcal{C}[\lambda x.M]]\!]$. We consider the applicative context $\mathcal{C}' \triangleq \mathcal{C}[\lambda x.[\cdot]]$. Then, we have $\sum[\![\mathcal{C}'[N]]\!] = \sum[\![\mathcal{C}[\lambda x.N]]\!] < \sum[\![\mathcal{C}[\lambda x.M]]\!] = \sum[\![\mathcal{C}'[M]]\!]$. Therefore, $M \leq_{\text{app}} N$ does not hold.

Let us now prove point (2). Suppose that $M \leq_{\mathrm{cxt}} N$ does not hold. Then, there exists $\mathcal{C} \in \mathsf{C}\Lambda_{\oplus}$ such that $\sum[\![\mathcal{C}[N]]\!] < \sum[\![\mathcal{C}[M]]\!]$. We consider the context $\mathcal{C}' \triangleq \mathcal{C}[[\cdot]x]$. By applying Corollary 116 twice and Lemma 99.(2), we can conclude $\sum[\![\mathcal{C}'[\lambda x.N]]\!] = \sum[\![\mathcal{C}[(\lambda x.N)x]]\!] = \sum[\![\mathcal{C}[N]]\!] < \sum[\![\mathcal{C}[M]]\!] = \sum[\![\mathcal{C}[(\lambda x.M)x]]\!] = \sum[\![\mathcal{C}'[\lambda x.M]]\!]$. Hence, $\lambda x.M \leq_{\mathrm{cxt}} \lambda x.N$ does not hold.

Last, we prove point (3). Suppose $M \leq_{\mathrm{cxt}} N$ and let $\mathcal{C} \in \mathsf{C}\Lambda_{\oplus}$. By defining $\mathcal{C}' \triangleq \mathcal{C}[[\cdot]L]$ we have $\sum[\![\mathcal{C}[ML]]\!] = \sum[\![\mathcal{C}'[M]]\!] \leq \sum[\![\mathcal{C}'[N]]\!] = \sum[\![\mathcal{C}[NL]]\!]$. Therefore, $ML \leq_{\mathrm{cxt}} NL$. □

In order to simplify the proof of Context Lemma, we shall adopt a slightly more general notion of context, allowing multiple holes.

Definition 82 (Generalized contexts)**.** Let $\mathcal{V}$ be a denumerable set of variables. A *generalized context* of $\Lambda_{\oplus}$ is a term containing holes $[\cdot]$, generated by the following grammar:

$$\mathcal{C} := x \mid [\cdot] \mid \lambda x.\mathcal{C} \mid \mathcal{C}\mathcal{C} \mid \mathcal{C} \oplus \mathcal{C} \tag{5.5}$$

where $x \in \mathcal{V}$. We denote by $\mathsf{G}\Lambda_{\oplus}$ the set of all generalized contexts. If $\mathcal{C} \in \mathsf{G}\Lambda_{\oplus}$ and $M \in \Lambda_{\oplus}$, then $\mathcal{C}[M]$ denotes a term obtained by substituting every hole in $\mathcal{C}$ with M allowing the possible capture of free variables of M.

Context lemma says that if two programs are distinguishable by some context then there is some *applicative context* that distinguish them.

Lemma 118. *Let $M, N \in \Lambda^{\emptyset}_{\oplus}$ be such that $M \leq_{\mathrm{app}} N$. Then $\sum[\![\mathcal{C}[M]]\!] \leq \sum[\![\mathcal{C}[N]]\!]$, for all $\mathcal{C} \in \mathsf{G}\Lambda_{\oplus}$.*

Proof. By Theorem 115, it is enough to show that, for all $n \in \mathbb{N}$ and for all contexts $\mathcal{C} \in \mathsf{G}\Lambda_{\oplus}$:

$$\sum_{H \in \mathrm{HNF}} \mathcal{H}^{n}(\mathcal{C}[M], H) \leq \sum_{H \in \mathrm{HNF}} \mathcal{H}^{\infty}(\mathcal{C}[N], H). \tag{5.6}$$

In what follows, we write $\sum \mathcal{H}^{n}(\mathcal{C}[M])$ (resp. $\sum \mathcal{H}^{\infty}(\mathcal{C}[M])$) in place of $\sum_{H \in \mathrm{HNF}} \mathcal{H}^{n}(\mathcal{C}[M], H)$ (resp. $\sum_{H \in \mathrm{HNF}} \mathcal{H}^{\infty}(\mathcal{C}[M], H)$). The proof is by induction on $(n, |\mathcal{C}|)$, where $n \in \mathbb{N}$ and $|\mathcal{C}|$ is the size of $\mathcal{C} \in \mathsf{G}\Lambda_{\oplus}$, i.e. the number of nodes in the syntax tree of $\mathcal{C}$. First, note that $\mathcal{C}$ must be of the form $\mathcal{C}_0\mathcal{C}_1 \ldots \mathcal{C}_k$, for some $k \in \mathbb{N}$. We have several cases depending on the structure of $\mathcal{C}_0$:

(a) $\mathcal{C}_0 = x$ then both $\mathcal{C}[M]$ and $\mathcal{C}[N]$ are head normal forms, and the inequation in (5.6) is straightforward.

(b) If $\mathcal{C}_0 = \lambda x.\mathcal{C}'$ then we have two cases:

   (i) If $k = 0$ then, by Lemma 109.(2) and by induction hypothesis, $\sum \mathcal{H}^{n}(\lambda x.\mathcal{C}'[M]) = \sum \mathcal{H}^{n}(\mathcal{C}'[M]) \leq \sum \mathcal{H}^{\infty}(\mathcal{C}'[N]) = \sum \mathcal{H}^{\infty}(\lambda x.\mathcal{C}'[N])$.

   (ii) For $k > 0$ we have two cases depending on $n \in \mathbb{N}$. If $n = 0$ then Lemma 109.(1) implies $\sum \mathcal{H}^{0}((\lambda x.\mathcal{C}'[M])\mathcal{C}_1[M] \ldots \mathcal{C}_k[M]) = 0$. Otherwise, by Lemma 109.(1) and by using the induction hypothesis, we have:

$$\begin{aligned}\sum \mathcal{H}^{n}((\lambda x.\mathcal{C}'[M])\mathcal{C}_1[M] \ldots \mathcal{C}_k[M]) &= \sum \mathcal{H}^{n-1}(((\mathcal{C}'[M])[\mathcal{C}_1[M]/x])\mathcal{C}_2[M] \ldots \mathcal{C}_k[M]) \\ &\leq \sum \mathcal{H}^{\infty}(((\mathcal{C}'[N])[\mathcal{C}_1[N]/x])\mathcal{C}_2[N] \ldots \mathcal{C}_k[N]) \\ &= \sum \mathcal{H}^{\infty}((\lambda x.\mathcal{C}'[N])\mathcal{C}_1[N] \ldots \mathcal{C}_k[N]).\end{aligned}$$

(c) If $\mathcal{C}_0 = \mathcal{C}' \oplus \mathcal{C}''$, then we have two cases depending on $n \in \mathbb{N}$. If $n = 0$, Lemma 109.(1) implies $\sum \mathcal{H}^n((\mathcal{C}'[M] \oplus \mathcal{C}''[M])\mathcal{C}_1[M] \ldots \mathcal{C}_k[M]) = 0$. Otherwise, by using the induction hypothesis and by Lemma 109.(1), we have:

$$\begin{aligned}\sum \mathcal{H}^n((\mathcal{C}'[M] \oplus \mathcal{C}''[M])\mathcal{C}_1[M] \ldots \mathcal{C}_k[M]) &= \frac{1}{2} \cdot \sum \mathcal{H}^{n-1}(\mathcal{C}'[M]\mathcal{C}_1[M] \ldots \mathcal{C}_k[M]) \\ &\quad + \frac{1}{2} \cdot \sum \mathcal{H}^{n-1}(\mathcal{C}''[M]\mathcal{C}_1[M] \ldots \mathcal{C}_k[M]) \\ &\le \frac{1}{2} \cdot \sum \mathcal{H}^{\infty}(\mathcal{C}'[N]\mathcal{C}_1[N] \ldots \mathcal{C}_k[N]) \\ &\quad + \frac{1}{2} \cdot \sum \mathcal{H}^{\infty}(\mathcal{C}''[N]\mathcal{C}_1[N] \ldots \mathcal{C}_k[N]) \\ &= \sum \mathcal{H}^{\infty}((\mathcal{C}'[N] \oplus \mathcal{C}''[N])\mathcal{C}_1[N] \ldots \mathcal{C}_k[N]).\end{aligned}$$

(d) The last case is when $\mathcal{C}_0 = [\cdot]$. First, note that $M = M_0 \ldots M_h$ for some $h \in \mathbb{N}$. Since M is closed, we can assume that $M_0 = \lambda x.M_0'$ is an abstraction. We apply Case (b) to the context $(\lambda x.M_0')M_1 \ldots M_h\mathcal{C}_1 \ldots \mathcal{C}_k$, and we have $\sum \mathcal{H}^n(M_0M_1 \ldots M_h\mathcal{C}_1[M] \ldots \mathcal{C}_k[M]) \le \sum \mathcal{H}^{\infty}(M_0M_1 \ldots M_h\mathcal{C}_1[N] \ldots \mathcal{C}_k[N])$. Since it holds that $M \le_{\text{app}} N$, we can conclude $\sum \mathcal{H}^{\infty}(M\mathcal{C}_1[N] \ldots \mathcal{C}_k[N]) \le \sum \mathcal{H}^{\infty}(N\mathcal{C}_1[N] \ldots \mathcal{C}_k[N])$. □

Lemma 119 (Context Lemma). *Let $M, N \in \Lambda_{\oplus}$:*

(1) $M \le_{\text{cxt}} N$ if and only if $M \le_{\text{app}} N$;

(2) $M =_{\text{cxt}} N$ if and only if $M =_{\text{app}} N$.

Proof. Point (2) follows directly from point (1). Lemma 118 gives us point (1) for $M, N \in \Lambda_{\oplus}^{\emptyset}$. We extend it to open terms as follows. If $M, N \in \Lambda_{\oplus}^{\{x_1,\ldots,x_n\}}$, then:

$$\begin{aligned}M \le_{\text{app}} N &\Rightarrow \lambda x_1 \ldots x_n.M \le_{\text{app}} \lambda x_1 \ldots x_n.N && \text{Lem. 117.(1)}\\ &\Rightarrow \lambda x_1 \ldots x_n.M \le_{\text{cxt}} \lambda x_1 \ldots x_n.N \\ &\Rightarrow M \le_{\text{cxt}} N. && \text{Lem. 117.(2)}\end{aligned}$$

This concludes the proof. □

5.3.2 The Soundness Theorem

Let us recall that, given $X \subseteq \text{HNF}$, $\precsim(X)$ denotes the image of X under $\precsim$. We start with some preliminary lemmas.

Lemma 120. *Let $H, H' \in \text{HNF}^{\{x\}}$. The following are equivalent statements:*

(1) $\lambda x.H \precsim \lambda x.H'$;

(2) $\nu x.H \precsim \nu x.H'$;

(3) $\forall P \in \Lambda_{\oplus}^{\emptyset},\ H[P/x] \precsim H'[P/x]$.

Proof. Let us first show that point (1) implies point (2). By Proposition 104, if $\lambda x.H \precsim \lambda x.H'$ then:

$$1 = \mathcal{P}_{\oplus}(\lambda x.H, \tau, \{\nu x.H\}) \le \mathcal{P}_{\oplus}(\lambda x.H', \tau, \precsim(\nu x.H)).$$

Hence, $\mathcal{P}_{\oplus}(\lambda x.H', \tau, \precsim(\nu x.H)) = 1$, so that $\nu x.H \precsim \nu x.H'$. To prove that point (2) implies point (3), if $\nu x.H \precsim \nu x.H'$ then, by Proposition 104, we have:

$$1 = \mathcal{P}_{\oplus}(\nu x.H, P, \{H[P/x]\}) \leq \mathcal{P}_{\oplus}(\nu x.H', P, \precsim(H[P/x])),$$

for all $P \in \Lambda_{\oplus}^{\emptyset}$. Hence, $\mathcal{P}_{\oplus}(\nu x.H', P, \precsim(H[P/x])) = 1$, so that $H[P/x] \precsim H'[P/x]$. We now prove that point (3) implies point (2). Let us consider the relation $\mathcal{R}$ defined by:

$$\{(\nu x.H, \nu x.H') \in \widetilde{\mathrm{HNF}} \times \widetilde{\mathrm{HNF}} \mid \forall P \in \Lambda_{\oplus}^{\emptyset},\ H[P/x] \precsim H'[P/x]\} \cup \precsim$$

Clearly, $\mathcal{R}$ is a preorder because $\precsim$ is. Now, if we show that $\mathcal{R}$ is a simulation then $\mathcal{R} \subseteq \precsim$, so that $\nu x.H \precsim \nu x.H'$ holds whenever $H[P/x] \precsim H'[P/x]$ for all $P \in \Lambda_{\oplus}^{\emptyset}$. The only interesting case is $\nu x.H\ \mathcal{R}\ \nu x.H'$. Let $P \in \Lambda_{\oplus}^{\emptyset}$. By definition, we have $H[P/x] \precsim H'[P/x]$, so that:

$$\begin{aligned}\mathcal{P}_{\oplus}(\nu x.H, P, \{H[P/x]\}) &\leq \mathcal{P}_{\oplus}(\nu x.H', P, \precsim(\{H[P/x]\})) \\ &\leq \mathcal{P}_{\oplus}(\nu x.H', P, \mathcal{R}(\{H[P/x]\})).\end{aligned}$$

Finally, we prove that point (2) implies point (1). Let us consider the following relation:

$$\mathcal{R} \triangleq \{(\lambda x.H, \lambda x.H') \in \mathrm{HNF} \times \mathrm{HNF} \mid \nu x.H \precsim \nu x.H'\} \cup \precsim.$$

It is a preorder because $\precsim$ is. Now, if we show that $\mathcal{R}$ is a simulation then $\mathcal{R} \subseteq \precsim$, so that $\lambda x.H \precsim \lambda x.H'$ whenever $\nu x.H \precsim \nu x.H'$. The only interesting case is $\lambda x.H\ \mathcal{R}\ \lambda x.H'$. By definition, we have $\nu x.H \precsim \nu x.H'$, so that $\mathcal{P}_{\oplus}(\lambda x.H, \tau, \{\nu x.H\}) \leq \mathcal{P}_{\oplus}(\lambda x.H', \tau, \precsim(\{\nu x.H\})) \leq \mathcal{P}_{\oplus}(\lambda x.H', \tau, \mathcal{R}(\{\nu x.H\}))$. □

Given $X \subseteq \mathrm{HNF}^{\{x\}}$, $\nu x.\precsim(X)$ denotes the set of distinguished hnfs $\{\nu x.H \mid H \in \precsim(X)\}$, while $\lambda x.\precsim(X)$ denotes the set of terms $\{\lambda x.M \mid M \in \precsim(X)\}$.

Lemma 121. *Let $X \subseteq \mathrm{HNF}^{\{x\}}$:*

$$\begin{aligned}\precsim(\lambda x.X) \cap \mathrm{HNF}^{\emptyset} &= \lambda x.\precsim(X) \cap \mathrm{HNF}^{\emptyset}, \\ \precsim(\nu x.X) &= \nu x.\precsim(X).\end{aligned}$$

Proof. Let us prove the first equation. For all $\lambda x.H \in \mathrm{HNF}^{\emptyset}$:

$$\begin{aligned}\lambda x.H \in \precsim(\lambda x.X) &\Leftrightarrow \exists H' \in X,\ \lambda x.H' \precsim \lambda x.H \\ &\Leftrightarrow \exists H' \in X,\ H' \precsim H && \text{Def. 79} \\ &\Leftrightarrow \lambda x.H \in \lambda x.\precsim(X).\end{aligned}$$

Concerning the second equation, first note that $\precsim(\nu x.X)$ contains only distinguished head normal forms. Indeed, suppose $M \in \precsim(\nu x.X)$ for some term $M \in \Lambda_{\oplus}^{\emptyset}$. Then, there exists $H \in X$ such that $\nu x.H \precsim M$. By Proposition 104, we would have $1 = \mathcal{P}_{\oplus}(\nu x.H, P, \{H[P/x]\}) \leq \mathcal{P}_{\oplus}(M, P, \precsim(\{H[P/x]\})) = 0$. Now, for all $\nu x.H \in \widetilde{\mathrm{HNF}}$, we have:

$$\begin{aligned}\nu x.H \in \precsim(\nu x.X) &\Leftrightarrow \exists H' \in X,\ \nu x.H' \precsim \nu x.H \\ &\Leftrightarrow \exists H' \in X,\ \lambda x.H' \precsim \lambda x.H && \text{Lem. 120} \\ &\Leftrightarrow \exists H' \in X,\ H' \precsim H && \text{Def. 79} \\ &\Leftrightarrow \nu x.H \in \nu x.\precsim(X).\end{aligned}$$

This concludes the proof. □

Lemma 122. *Let $M, N \in \Lambda_{\oplus}^{\emptyset}$. For all $X \subseteq \mathrm{HNF}^{\emptyset}$, $[\![M]\!](X) \leq [\![N]\!](\precsim(X))$ if and only if $M \precsim N$.*

Proof. The right-to-left direction follows from Proposition 104. Concerning the converse, we define $\mathcal{R}$ as:

$$\{(P, Q) \in \Lambda_{\oplus}^{\emptyset} \times \Lambda_{\oplus}^{\emptyset} \mid \forall X \subseteq \mathrm{HNF}^{\emptyset},\ [\![P]\!](X) \leq [\![Q]\!](\precsim(X))\} \cup \precsim$$

If we prove that $\mathcal{R}$ is a probabilistic simulation then $\mathcal{R} \subseteq \precsim$, so that $M \precsim N$ whenever $[\![M]\!](X) \leq [\![N]\!](\precsim(X))$, for all $X \subseteq \mathrm{HNF}^{\emptyset}$. So, let us first prove that $\mathcal{R}$ is a preorder. Clearly, $\mathcal{R}$ is reflexive. To prove transitivity, let $P, Q, L \in \Lambda_{\oplus}^{\emptyset}$ be such that $P\ \mathcal{R}\ L$ and $L\ \mathcal{R}\ Q$. By Proposition 104, $\precsim$ is transitive. It follows that, for all $X \subseteq \mathrm{HNF}^{\emptyset}$:

$$[\![P]\!](X) \leq [\![L]\!](\precsim(X)) \leq [\![Q]\!](\precsim(\precsim(X))) \leq [\![Q]\!](\precsim(X))$$

Now, let $P, Q \in \Lambda_{\oplus}^{\emptyset}$ be such that $P\ \mathcal{R}\ Q$, and let $X \subseteq \mathrm{HNF}^{\{x\}}$. We have:

$$\begin{aligned}
\mathcal{P}_{\oplus}(P, \tau, \nu x.X) &= [\![P]\!](\lambda x.X) & \\
&\leq [\![Q]\!](\precsim(\lambda x.X)) & \\
&= [\![Q]\!](\precsim(\lambda x.X) \cap \mathrm{HNF}^{\emptyset}) & \text{since } Q \in \Lambda_{\oplus}^{\emptyset} \\
&= [\![Q]\!](\lambda x.\precsim(X) \cap \mathrm{HNF}^{\emptyset}) & \text{Lem. 121} \\
&= [\![Q]\!](\lambda x.\precsim(X)) & Q \in \Lambda_{\oplus}^{\emptyset} \\
&= \mathcal{P}_{\oplus}(Q, \tau, \nu x.\precsim(X)) & \\
&= \mathcal{P}_{\oplus}(Q, \tau, \precsim(\nu x.X)) & \text{Lem. 121} \\
&\leq \mathcal{P}_{\oplus}(Q, \tau, \mathcal{R}(\nu x.X)). &
\end{aligned}$$

Therefore, $\mathcal{R}$ is a probabilistic simulation. □

The forthcoming Lemma 124 describes the applicative behaviour of $\precsim$ and it requires an auxiliary result about the so-called "probabilistic assignments". Probabilistic assignments were first introduced in this setting by [27] to prove the soundness of PAS in the lazy cbn.

Definition 83 (Probabilistic assignments)**.** A *probabilistic assignment* is defined as a pair $(\{p_i\}_{1\leq i\leq n}, \{r_I\}_{I\subseteq\{1,\dots,n\}})$, with all p_i, r_I in $[0,1]$, such that, for all $I \subseteq \{1, \dots, n\}$:

$$\sum_{i\in I} p_i \leq \sum_{\substack{J\subseteq\{1,\dots,n\} \\ \text{s.t. } J\cap I\neq\emptyset}} r_J. \tag{5.7}$$

Lemma 123 ([27])**.** *Let $(\{p_i\}_{1\leq i\leq n}, \{r_I\}_{I\subseteq\{1,\dots,n\}})$ be a probabilistic assignment. Then for every $I \subseteq \{1, \dots, n\}$ and for every $k \in I$ there is $s_{k,I} \in [0,1]$ such that:*

(1) $\forall j \in \{1, \dots, n\}$: $p_j \leq \sum_{\substack{J\subseteq\{1,\dots,n\} \\ s.t.\ j\in J}} s_{j,J} \cdot r_J$;

(2) $\forall J \subseteq \{1, \dots, n\}$: $\sum_{\substack{j\in\{1,\dots,n\} \\ s.t.\ j\in J}} s_{j,J} \leq 1$.

Following essentially the same ideas of [27], we shall use the above property to decompose and recombine distributions in the proof of the following lemma.

Lemma 124 (Key Lemma)**.** *Let $M, N \in \Lambda_{\oplus}^{\emptyset}$. If $M \precsim N$ then, for all $P \in \Lambda_{\oplus}^{\emptyset}$, $MP \precsim NP$.*

Proof. By Lemma 122 it suffices to prove that, for all $X \subseteq \mathrm{HNF}^{\emptyset}$, $[\![MP]\!](X) \leq [\![NP]\!](\precsim(X))$. This amounts to show that, for all $\mathscr{D}$ such that $MP \Downarrow \mathscr{D}$, it holds that $\mathscr{D}(X) \leq [\![NP]\!](\precsim(X))$. This is trivial when $\mathscr{D} = \bot$, so that we can assume that the last rule in the derivation of $MP \Downarrow \mathscr{D}$ is the following:

$$\frac{M \Downarrow \mathscr{E} \qquad \{H[P/x] \Downarrow \mathscr{F}_{H,P}\}_{\lambda x.H \in \mathrm{supp}(\mathscr{E})}}{MP \Downarrow \sum_{\lambda x.H \in \mathrm{supp}(\mathscr{E})} \mathscr{E}(\lambda x.H) \cdot \mathscr{F}_{H,P}}\ s4$$

Since $\mathscr{E}$ is a finite distribution, $\mathscr{D}$ is a sum of finitely many summands. Let $\mathrm{supp}(\mathscr{E})$ be $\{\lambda z.H_1, \ldots, \lambda z.H_n\} \subseteq \mathrm{HNF}^{\emptyset}$. We define the pair $(\{p_i\}_{1 \leq i \leq n}, \{r_I\}_{I \subseteq \{1,\ldots,n\}})$ as follows:

(a) $\forall i \leq n$: $p_i \triangleq \mathscr{E}(\lambda z.H_i)$;

(b) $\forall I \subseteq \{1, \ldots, n\}$:
$$r_I \triangleq \sum_{\substack{\lambda z.H' \text{ s.t.} \\ \{i \leq n \ \mid \ \lambda z.H' \in \precsim(\lambda z.H_i)\} = I}} [\![N]\!](\lambda z.H').$$

Let us show that $(\{p_i\}_{1 \leq i \leq n}, \{r_I\}_{I \subseteq \{1,\ldots,n\}})$ is a probabilistic assignment by proving that the condition in (5.7) holds. First, from $M \precsim N$ and by Lemma 122, we have $\mathscr{E}(\bigcup_{i \in I}\{\lambda z.H_i\}) \leq [\![N]\!](\bigcup_{i \in I} \precsim(\lambda z.H_i))$. Moreover, $[\![N]\!](\bigcup_{i \in I} \precsim(\lambda z.H_i)) = [\![N]\!](\bigcup_{i \in I} \precsim(\lambda z.H_i) \cap \mathrm{HNF}^{\emptyset})$ because $N \in \Lambda_{\oplus}^{\emptyset}$. Finally, for all $I \subseteq \{1, \ldots, n\}$:

$$\begin{aligned}\sum_{i \in I} p_i &= \sum_{i \in I} \mathscr{E}(\lambda z.H_i) = \mathscr{E}(\bigcup_{i \in I}\{\lambda z.H_i\}) \leq [\![N]\!](\bigcup_{i \in I} \precsim(\lambda z.H_i)) = [\![N]\!](\bigcup_{i \in I} \precsim(\lambda z.H_i) \cap \mathrm{HNF}^{\emptyset}) \\ &= \sum_{\substack{\lambda z.H' \in \\ \bigcup_{i \in I} \precsim(\lambda z.H_i)}} [\![N]\!](\lambda z.H') \leq \sum_{\substack{J \subseteq \{1,\ldots,n\} \\ \text{s.t. } J \cap I \neq \emptyset}} r_J \,.\end{aligned}$$

By applying Lemma 123, for all $I = \{1, \ldots, n\}$ and for every $k \in I$ there exists $h_{k,I} \in [0, 1]$ such that:

$$\forall j \leq n: \qquad p_j \leq \sum_{\substack{J \subseteq \{1,\ldots,n\} \\ \text{s.t. } j \in J}} h_{j,J} \cdot r_J\,; \tag{5.8}$$

$$\forall J \subseteq \{1, \ldots, n\}: \qquad 1 \geq \sum_{\substack{j \in \{1,\ldots,n\} \\ \text{s.t. } j \in J}} h_{j,J}\,. \tag{5.9}$$

We now show that, for all $\lambda z.H' \in \bigcup_{i \in I} \precsim(\lambda z.H_i)$, there exist n real numbers $s_1^{H'}, \ldots, s_n^{H'}$ such that:

$$\forall i \leq n: \qquad \mathscr{E}(\lambda z.H_i) \leq \sum_{\lambda z.H' \in \precsim(\lambda z.H_i)} s_i^{H'}\,; \tag{5.10}$$

$$\forall \lambda z.H' \in \bigcup_{i \in I} \precsim(\lambda z.H_i): \qquad [\![N]\!](\lambda z.H') \geq \sum_{i=1}^{n} s_i^{H'}\,. \tag{5.11}$$

For all $i \leq n$ and for all $\lambda z.H' \in \precsim(\lambda z.H_i)$, we set:

$$s_i^{H'} \triangleq h_{i,\{k \leq n \ \mid \ \lambda z.H' \in \precsim(\lambda z.H_k)\}} \cdot [\![N]\!](\lambda z.H').$$

Concerning the inequation in (5.10), by using the inequation in (5.8) we have, for all $i \leq n$:

$$\begin{aligned}
\mathscr{E}(\lambda z.H_i) &\leq \sum_{\substack{I\subseteq\{1,\ldots n\}\\ \text{s.t. } i\in I}} h_{i,I}\cdot r_I \\
&= \sum_{\substack{I\subseteq\{1,\ldots n\}\\ \text{s.t. } i\in I}} h_{i,I}\cdot \left(\sum_{\substack{\lambda z.H' \text{ s.t.}\\ \{k\leq n \mid \lambda z.H'\in\precsim(\lambda z.H_k)\}=I}} [\![N]\!](\lambda z.H')\right) \\
&= \sum_{\lambda z.H'\in\precsim(\lambda z.H_i)} h_{i,\{k\leq n \mid \lambda z.H'\in\precsim(\lambda z.H_k)\}}\cdot [\![N]\!](\lambda z.H') = \sum_{\lambda z.H'\in\precsim(\lambda z.H_i)} s_i^{H'}
\end{aligned}$$

As for the inequation in (5.11), by using the inequation in (5.9) we have, for all $\lambda z.H' \in \bigcup_{i\in I} \precsim(\lambda z.H_i)$:

$$\sum_{i=1}^{n} s_i^{H'} = \sum_{i=1}^{n} h_{i,\{k\leq n \mid \lambda z.H'\in\precsim(\lambda z.H_k)\}}\cdot [\![N]\!](\lambda z.H') \leq [\![N]\!](\lambda z.H')$$

We are now able to prove that $\mathscr{D}(X) \leq [\![NP]\!](\precsim(X))$. First, by applying Lemma 120 and Lemma 122, for all $i \leq n$, for all $\lambda z.H' \in \precsim(\lambda z.H_i)$, for all $P \in \Lambda_\oplus^\emptyset$, and for all $X \subseteq \mathrm{HNF}^\emptyset$:

$$\mathscr{F}_{H_i,P}(X) \leq [\![H_i[P/x]]\!](X) \leq [\![H'[P/x]]\!](\precsim(X)) \tag{5.12}$$

Therefore, for all $X \subseteq \mathrm{HNF}^\emptyset$:

$$\begin{aligned}
\mathscr{D}(X) &\leq \sum_{i=1}^{n}\left(\sum_{\lambda z.H'\in\precsim(\lambda z.H_i)} s_i^{H'}\right)\cdot \mathscr{F}_{H_i,P}(X) = && \text{by (5.10)}\\
&= \sum_{i=1}^{n}\sum_{\lambda z.H'\in\precsim(\lambda z.H_i)} s_i^{H'}\cdot \mathscr{F}_{H_i,P}(X) \\
&\leq \sum_{i=1}^{n}\sum_{\lambda z.H'\in\precsim(\lambda z.H_i)} s_i^{H'}\cdot [\![H'[P/z]]\!](\precsim(X)) && \text{by (5.12)}\\
&\leq \sum_{i=1}^{n}\sum_{\substack{\lambda z.H'\in\\ \bigcup_{i=1}^{n}\precsim(\lambda z.H_i)}} s_i^{H'}\cdot [\![H'[P/z]]\!](\precsim(X)) \\
&\leq \sum_{\substack{\lambda z.H'\in\\ \bigcup_{i=1}^{n}\precsim(\lambda z.H_i)}} \left(\sum_{i=1}^{n} s_i^{H'}\right)\cdot [\![H'[P/z]]\!](\precsim(X)) \\
&\leq \sum_{\substack{\lambda z.H'\in\\ \bigcup_{i=1}^{n}\precsim(\lambda z.H_i)}} [\![N]\!](\lambda z.H')\cdot [\![H'[P/z]]\!](\precsim(X)) && \text{by (5.11)}\\
&\leq \sum_{\lambda z.H'\in\,\mathrm{supp}([\![N]\!])} [\![N]\!](\lambda z.H')\cdot [\![H'[P/z]]\!](\precsim(X)) = [\![NP]\!](\precsim(X)) && \text{Prop. 97.(1)}
\end{aligned}$$

and hence $\mathscr{D}(X) \leq [\![NP]\!](\precsim(X))$. ☐

Lemma 125. *Let $M, N \in \Lambda_\oplus^\emptyset$. If $M \precsim N$ then $M \leq_{\mathrm{app}} N$.*

Proof. We have to show that $M \precsim N$ implies $\sum[\![MP_1 \dots P_n]\!] \leq \sum[\![NP_1 \dots P_n]\!]$, for any sequence $P_1, \dots, P_n \in \Lambda_\oplus^\emptyset$. The proof is by induction on n. If $n = 0$ then, from $M \precsim N$ and by Lemma 122, we have:

$$\sum[\![M]\!] = [\![M]\!](\mathrm{HNF}^\emptyset) \leq [\![N]\!](\precsim(\mathrm{HNF}^\emptyset)) = [\![N]\!](\mathrm{HNF}^\emptyset) = \sum[\![N]\!].$$

If $n > 0$ then $MP_1 \precsim NP_1$ by Lemma 124. We conclude by applying the induction hypothesis on MP_1 and NP_1. □

We are now able to prove that PAS (resp. PAB) is sound with respect to context preorder (resp. context equivalence), a first step toward full abstraction.

Theorem 126 (Soundness)**.** *Let $M, N \in \Lambda_\oplus$:*

(1) $M \precsim N$ implies $M \leq_{\mathrm{cxt}} N$;

(2) $M \sim N$ implies $M =_{\mathrm{cxt}} N$.

Proof. Point (2) follows from point (1) since $\sim \, = \, \precsim \cap (\precsim)^{op}$ (Proposition 107) and since $=_{\mathrm{cxt}}$ is $\leq_{\mathrm{cxt}} \cap (\leq_{\mathrm{cxt}})^{op}$. Concerning point (1), we first prove it for closed terms. So, let $M, N \in \Lambda_\oplus^\emptyset$ be such that $M \precsim N$. By Lemma 125, it holds that $M \leq_{\mathrm{app}} N$. By Lemma 119, this implies $M \leq_{\mathrm{cxt}} N$. Now, let $M, N \in \Lambda_\oplus^{\{x_1,\dots,x_n\}}$ be such that $M \precsim N$. From Definition 79, we have that $\lambda x_1 \dots x_n.M \precsim \lambda x_1 \dots x_n.N$. Because these are closed terms, we obtain $\lambda x_1 \dots x_n.M \leq_{\mathrm{cxt}} \lambda x_1 \dots x_n.N$. By repeatedly applying Lemma 117.(2), we conclude $M \leq_{\mathrm{cxt}} N$. □

5.4 Full abstraction

In this section we prove that PAB is complete, and hence fully abstract for the head reduction (Theorem 134). Moreover, by using the Context Lemma, we give a counterexample to the completeness for PAS (see (5.16)) and we discuss how this property could be restored by adding Plotkin's "parallel disjunction" [76] to $\Lambda_\oplus$, as done in [23]. We motivate our conjecture by showing that the counterexample no longer applies in the extended language, if endowed with a suitable operational semantics respecting some invariance properties.

Concerning the proof of the completeness for PAB, this is usually achieved by transforming PAB into a testing semantics defined by Larsen and Skou [58], which has been proved equivalent to probabilistic bisimulation by van Breugel et al. [92], and then by showing that every test is definable by a context in the language, see e.g. [22, 53]. This reasoning is not so simple to implement in our setting, since defining tests requires a kind of sampling primitive, that may or may not be representable in a call-by-name semantics, as already remarked in the introduction of this chapter.

Nonetheless, we succeed in following a different path, based on Leventis' Separation Theorem [60]. The idea is to prove that (a trivial extension of) the context equivalence is a probabilistic applicative bisimulation, hence contained in $\sim$ by Definition 76.(2). Basically, this boils down to checking that for any context equivalence class E of head normal forms and any $M =_{\mathrm{cxt}} N$, we have $[\![M]\!](E) = [\![N]\!](E)$ (Lemma 133). To prove this, we shall associate with every term a kind of infinitary, extensional normal forms, the so-called probabilistic Nakajima trees (Section 5.4.1). Using the Separation Theorem, stating that two terms M and N share the same Nakajima tree whenever they are context equivalent (here Theorem 129), we shall look at such trees as the representatives of the context equivalence classes. The missing ingredient will be then to show that the quantity $[\![M]\!](E)$ depends only on the Nakajima tree of M and that of E (Lemma 132), and this allows us conclude $[\![M]\!](E) = [\![N]\!](E)$ and hence the full abstraction result.

5.4.1 Probabilistic Nakajima trees

A *Böhm tree* [10] is a labelled tree describing a kind of infinitary normal form of any deterministic λ-term. In more details, the Böhm tree $BT(M)$ of a λ-term M can be given co-inductively as follows:

- if the head reduction of M terminates into the head normal form $\lambda x_1 \dots x_n.yM_1 \dots M_m$, then:

$$\begin{tikzcd}
 & & \lambda x_1 \dots x_n.y \arrow[dl, dash] \arrow[dr, dash] & \\
BT(M) \triangleq & BT(M_1) & \dots & BT(M_m)
\end{tikzcd}$$

  where $BT(M_1), \dots, BT(M_m)$ are the Böhm trees of the subterms $M_1, \dots, M_m$ of the head normal form of M;

- otherwise, the tree is a node labelled by $\mathbf{\Omega}$.

The notion of Böhm tree is not sufficient to characterize context equivalence because it lacks extensionality: the terms y and $\lambda z.yz$ have different Böhm trees and yet $y =_{\text{cxt}} \lambda z.yz$ holds. To recover extensionality, we need the so-called *Nakajima trees* [74], which are infinitely η-expanded representations of the Böhm trees. The Nakajima tree $BT^\eta(H)$ of a head normal form $H = \lambda x_1 \dots x_n.yM_1 \dots M_m$ is the infinitely branching tree:

$$\begin{tikzcd}
 & & \lambda x_1 \dots x_n x_{n+1} \dots .y \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & & \\
BT^\eta(H) \triangleq & BT^\eta(M_1) \quad \dots & BT^\eta(M_m) & BT^\eta(x_{n+1}) \quad \dots &
\end{tikzcd}$$

where $x_1 \dots x_n x_{n+1} \dots$ is an infinite sequence of pairwise distinct variables and, for $i > n$, the x_i's are fresh.

Nakajima trees represent infinitary η-long head normal forms. Every head normal form $H = \lambda x_1 \dots x_n.yM_1 \dots M_m$ η-expands into $\lambda x_1 \dots x_{n+k}.yM_1 \dots M_m x_{n+1} \dots x_{n+k}$ for any $k \in \mathbb{N}$ and $x_{n+1} \dots x_{n+k}$ fresh: Nakajima trees are, intuitively, the asymptotical representations of these η-expansions.

To generalize such a construction to probabilistic terms, we define by mutual recursion the tree associated with a head normal form and the tree of an arbitrary term M, the latter being a subprobability distribution over the trees of the head normal forms M reduces to. Hence, strictly speaking, a probabilistic Nakajima tree is not properly a tree.

Following Leventis [60] we shall give an inductive, "level-by-level" definition of the probabilistic Nakajima trees.

Definition 84 (Probabilistic Nakajima trees)**.** The set $\mathcal{PT}^\eta_\ell$ of *probabilistic Nakajima trees* with level at most $\ell \in \mathbb{N}$ is the set of subprobability distributions over *value Nakajima trees* $\mathcal{VT}^\eta_\ell$. These sets are defined by mutual recursion as follows:

$$\begin{aligned}
\mathcal{VT}^\eta_0 &\triangleq \emptyset & \mathcal{VT}^\eta_{\ell+1} &\triangleq \{\lambda x_1 x_2 \dots .y\, T_1, T_2, \dots \mid T_i \in \mathcal{PT}^\eta_\ell,\ \forall i \geq 1\} \\
\mathcal{PT}^\eta_0 &\triangleq \{\bot\} & \mathcal{PT}^\eta_{\ell+1} &\triangleq \{T : \mathcal{VT}^\eta_{\ell+1} \to [0,1] \mid \sum_{t \in \mathcal{VT}^\eta_{\ell+1}} T(t) \leq 1\}
\end{aligned}$$

where $\bot$ represents the distribution with empty support.

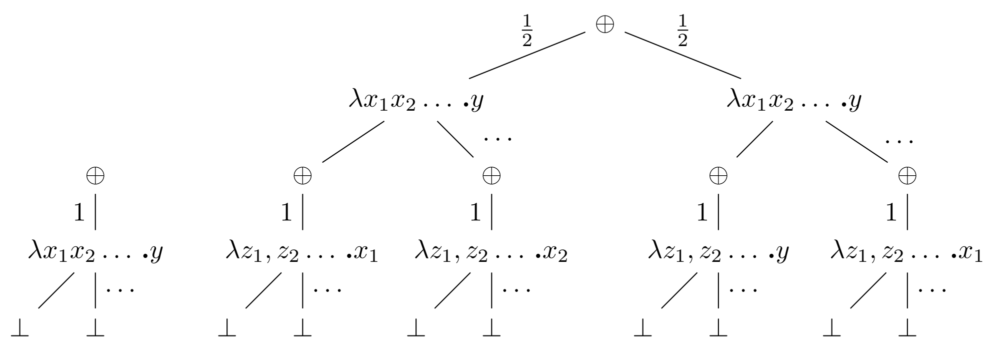
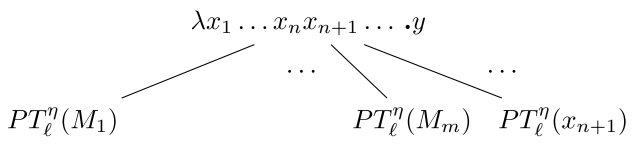

$\frac{1}{2}$
$\oplus$
$\frac{1}{2}$
$\lambda x_1 x_2 \ldots .y$
$\lambda x_1 x_2 \ldots .y$
$\ldots$
$\ldots$
$\oplus$
$\oplus$
$\oplus$
$\oplus$
$\oplus$
1
1
1
1
1
$\lambda x_1 x_2 \ldots .y$
$\lambda z_1, z_2 \ldots .x_1$
$\lambda z_1, z_2 \ldots .x_2$
$\lambda z_1, z_2 \ldots .y$
$\lambda z_1, z_2 \ldots .x_1$
$\ldots$
$\perp$

Figure 5.4: From left, the Nakajima trees $PT_1^\eta(\mathbf{\Theta}(\lambda f.(y \oplus yf)))$ and $PT_2^\eta(\mathbf{\Theta}(\lambda f.(y \oplus yf)))$.

Value Nakajima trees are ranged over by t, and probabilistic Nakajima trees are ranged over by T.

Definition 85 (Probabilistic Nakajima tree equality)**.** Let $\ell \in \mathbb{N}$. By mutual recursion we define a function $VT_{\ell+1}^\eta$ associating with each $H \in$ HNF its value Nakajima tree $VT_{\ell+1}^\eta(H)$ of level $\ell+1$, and a function PT_ℓ^η association with each $M \in \Lambda_\oplus$ its probabilistic Nakajima tree $PT_\ell^\eta(M)$ of level ℓ:

- if $H = \lambda x_1 \ldots x_n.yM_1 \ldots M_m$, then $VT_{\ell+1}^\eta(H)$ is:



  where $x_1 \ldots x_n x_{n+1} \ldots$ is an infinite sequence of pairwise distinct variables and, for $i > n$, the x_i's are fresh;

- $PT_\ell^\eta(M) \triangleq \begin{cases} t \mapsto \sum_{H \in (VT_\ell^\eta)^{-1}(t)} [\![M]\!](H) & \text{if } \ell > 0, \\ \perp & \text{otherwise.} \end{cases}$

We say that M and N *have the same Nakajima tree*, and we write $M =_{\mathrm{PT}^\eta} N$, if $PT_\ell^\eta(M) = PT_\ell^\eta(N)$ holds for all $\ell \in \mathbb{N}$.

Theorem 115 assures that the above definition based on the operational semantics $[\![\cdot]\!]$ given in Definition 71 is equivalent to the one given by Leventis in [60], based on the head reduction.

Example 31. Figure 5.4 depicts the Nakajima trees of level, respectively, 1 and 2 associated with term $\mathbf{\Theta}(\lambda f.(y \oplus yf))$, where $\mathbf{\Theta}$ is the Turing fixed-point combinator (Example 22). Distributions are represented by barycentric sums, depicted as $\oplus$ nodes whose outgoing edges are weighted by probabilities. Notice that the more the level ℓ increases, the more the top-level distribution's support grows.

Proposition 127 ([60])**.** *Let $M, N \in \Lambda_\oplus$. If $PT_\ell^\eta(M) = PT_\ell^\eta(N)$ for some $\ell \in \mathbb{N}$, then $PT_{\ell'}^\eta(M) = PT_{\ell'}^\eta(N)$ for all $\ell' \le \ell$.*

Proof. It suffices to prove by induction on $\ell \in \mathbb{N}$ that $PT^{\eta}_{\ell}(M) \neq PT^{\eta}_{\ell}(N)$ implies $PT^{\eta}_{\ell+1}(M) \neq PT^{\eta}_{\ell+1}(N)$. □

The next lemma shows that we can recover some informations about the shape of two terms from their Nakajima tree equivalence.

Lemma 128 ([61]). *Let $H = \lambda x_1 \ldots x_n.yM_1 \ldots M_m$ and $H' = \lambda x_1 \ldots x_{n'}.y'M'_1 \ldots M'_{m'}$ be two head normal forms, and let $\ell \geq 2$. Then $VT^{\eta}_{\ell}(H) = VT^{\eta}_{\ell}(H')$ implies both $y = y'$ and $n - m = n' - m'$.*

Proof. The fact $y = y'$ follows immediately from the definition of VT^{η}_{ℓ}. Concerning the second equality, one can assume $n = n'$ by expanding one of the two terms, since $n - m$ (resp. $n' - m'$) is invariant under η-expansion. Modulo α-equivalence, we can then restrict ourselves to consider the case of $H = \lambda x_1 \ldots x_n.yM_1 \ldots M_m$ and $H' = \lambda x_1 \ldots x_n.yM'_1 \ldots M'_{m'}$.

Suppose for the sake of contradiction, that $m > m'$. Then we should have $PT^{\eta}_{\ell-1}(M_{m'+1}) = PT^{\eta}_{\ell-1}(x_{n+1})$, were x_{n+1} is a fresh variable. In particular, it must be that $x_{n+1} \notin FV(M_{m'+1})$. Since $\ell - 1 > 0$, we have that $PT^{\eta}_{\ell-1}(x_{n+1})(t) = 1$ only if t is equal to:

$$\begin{tikzcd}
 & & \lambda z_1 z_2 \ldots .x_{n+1} \arrow[dll, dash] \arrow[dl, dash] & \ldots \\
PT^{\eta}_{\ell-2}(z_1) & PT^{\eta}_{\ell-2}(z_2) & &
\end{tikzcd}$$

otherwise $PT^{\eta}_{\ell-1}(x_{n+1})(t) = 0$. So, $PT^{\eta}_{\ell-1}(M_{m'+1}) = PT^{\eta}_{\ell-1}(x_{n+1})$ implies that $[\![M_{m'+1}]\!](H) > 0$ for some H having x_{n+1} as free variable, which is impossible since $x_{n+1} \notin FV(M_{m'+1})$. □

Theorem 129 (Separation [60]). *Let $M, N \in \Lambda_{\oplus}$. If $M =_{\mathrm{cxt}} N$ then $M =_{\mathrm{PT}^{\eta}} N$.*

5.4.2 The Completeness Theorem

In the previous subsection probabilistic Nakajima trees have been inductively presented by introducing "level-by-level" their finite representations. To recover the full quantitative information of a Nakajima tree we shall need a notion of approximation together with some general properties.

Definition 86 (ϵ-approximations). Let $r, r' \in \mathbb{R}$ and $\epsilon > 0$. We say that r *ϵ-approximates* r', and we write $r \approx_{\epsilon} r'$, if $|r - r'| < \epsilon$.

Fact 130. *Let $r, r', r'' \in \mathbb{R}$ and $\epsilon, \epsilon' > 0$. If $r \approx_{\epsilon} r'$ and $r' \approx_{\epsilon'} r''$ then $r \approx_{\epsilon+\epsilon'} r''$.*

Proof. We have $|r - r''| = |r - r' + r' - r''| \leq |r - r'| + |r' - r''| < \epsilon + \epsilon'$. □

Lemma 131. *Let $\{A_n\}_{n\in\mathbb{N}}$ be a descending chain of countable sets of positive real numbers satisfying $\sum_{r\in A_n} r < \infty$, for all $n \in \mathbb{N}$. Then:*

$$\sum_{r \,\in\, \bigcap_{n\in\mathbb{N}} A_n} r = \inf_{n\in\mathbb{N}} \left(\sum_{r \,\in\, A_n} r \right). \tag{5.13}$$

Proof. Henceforth, if A is a subset of real numbers, we let $\|A\|$ denote $\sum_{r\in A} r$. First, notice that it suffices to prove the following particular situation:

$$\text{if } \bigcap_{n\in\mathbb{N}} A_n = \emptyset \text{ then } \inf_{m\in\mathbb{N}} \|A_m\| = 0. \tag{5.14}$$

Let us show that the implication in (5.14) gives us the equation in (5.13). So, consider the chain $\{B_n\}_{n\in\mathbb{N}}$ defined by $B_n \triangleq A_n \setminus \bigcap_{m\in\mathbb{N}} A_m$. Since $\bigcap_{n\in\mathbb{N}} B_n = \emptyset$, then $\inf_{m\in\mathbb{N}} \|B_m\| = 0$ by (5.14). We have:

$$\begin{aligned}\|\bigcap_{n\in\mathbb{N}} A_n\| &= \|\bigcap_{n\in\mathbb{N}} A_n\| + \inf_{m\in\mathbb{N}} \|B_m\| \\ &= \inf_{m\in\mathbb{N}} (\|\bigcap_{n\in\mathbb{N}} A_n\| + \|B_m\|) \\ &= \inf_{m\in\mathbb{N}} (\|\bigcap_{n\in\mathbb{N}} A_n \cup B_m\|) \\ &= \inf_{m\in\mathbb{N}} \|A_m\|.\end{aligned}$$

So, let us prove (5.14) and suppose $\bigcap_{n\in\mathbb{N}} A_n = \emptyset$. Since $\{A_n\}_{n\in\mathbb{N}}$ is a descending chain such that $\forall n \in \mathbb{N}\ \|A_n\| < \infty$, we have that $\|A_n\|_{n\in\mathbb{N}}$ is a monotone decreasing sequence of positive real numbers. This means that $\lim_{n\to\infty} \|A_n\| = \inf_{n\in\mathbb{N}} \|A_n\|$. Thus, to prove the statement, it suffices to show that for all $\epsilon > 0$ there exists $k \in \mathbb{N}$ such that for all $m \geq k$ it holds that $\|A_m\| < \epsilon$. Now, given a A_n and $\epsilon > 0$, there always exists a finite subset of A_n, let us call it A_n^*, such that $\|A_n^*\| \approx_\epsilon \|A_n\|$. Moreover, since $\bigcap_{n\in\mathbb{N}} A_n = \emptyset$, for all $r \in A_n^*$ there exists a $n_r \in \mathbb{N}$ such that $r \notin A_{n_r}$. By considering A_k such that $k \triangleq \max_{r\in A_n^*} n_r$ we have $A_k \subseteq A_n \setminus A_n^*$. Hence, $\|A_k\| \leq \|A_n \setminus A_n^*\| = \|A_n\| - \|A_n^*\| < \epsilon$. □

A consequence of Theorem 129 is that for every context equivalence class $E \in \Lambda_\oplus^\emptyset / =_{\text{cxt}}$ and for every level $\ell \in \mathbb{N}$ there exists a *unique* value Nakajima tree t of that level such that $VT_\ell^\eta(H) = t$ for all $H \in E$. Let $t_{E,\ell}$ denote such a tree.

Lemma 132. *Let $M \in \Lambda_\oplus^\emptyset$ and $E \in \Lambda_\oplus^\emptyset / =_{\text{cxt}}$. Then:*

(1) $[\![M]\!](E) = \inf_{\ell\in\mathbb{N}} (PT_\ell^\eta(M)(t_{E,\ell}))$,

(2) $\forall\epsilon > 0\ \exists\ell \in \mathbb{N}\ \forall\ell' \geq \ell$: $[\![M]\!](E) \approx_\epsilon PT_{\ell'}^\eta(M)(t_{E,\ell'})$.

Proof. Let $E_{\mathsf{V}} \triangleq E \cap \text{HNF}^\emptyset$, notice that $[\![M]\!](E) = [\![M]\!](E_{\mathsf{V}})$. As for point (1), we have $H \in E_{\mathsf{V}}$ if and only if $\forall\ell \in \mathbb{N}\ VT_\ell^\eta(H) = t_{E,\ell}$ if and only if $\forall\ell \in \mathbb{N}\ H \in (VT_\ell^\eta)^{-1}(t_{E,\ell})$, so that $E_{\mathsf{V}} = \bigcap_{\ell\in\mathbb{N}} (VT_\ell^\eta)^{-1}(t_{E,\ell})$. Moreover, by Proposition 127, for all $\ell \in \mathbb{N}$ it holds that:

$$\begin{aligned}(VT_{\ell+1}^\eta)^{-1}(t_{E,\ell+1}) &= \{H \in \text{HNF}^\emptyset \mid VT_{\ell+1}^\eta(H) = t_{E,\ell+1}\} \\ &\subseteq \{H \in \text{HNF}^\emptyset \mid VT_\ell^\eta(H) = t_{E,\ell}\} \\ &= (VT_\ell^\eta)^{-1}(t_{E,\ell}).\end{aligned} \tag{5.15}$$

Therefore, $((VT_\ell^\eta)^{-1}(t_{E,\ell}))_{\ell\in\mathbb{N}}$ is a descending chain, so that $\{[\![M]\!](H) \mid H \in (VT_\ell^\eta)^{-1}(t_{E,\ell})\}_{\ell\in\mathbb{N}}$ is. Moreover, by definition we have $\sum_{H\in(VT_\ell^\eta)^{-1}(t_{E,\ell})} [\![M]\!](H) \leq \sum [\![M]\!] \leq 1$, for all $\ell \in \mathbb{N}$. Hence, by applying Lemma 131 and by definition of Nakajima tree equality, we have:

$$\begin{aligned}[\![M]\!](E) = \sum_{H\in E_{\mathsf{V}}} [\![M]\!](H) &= \sum_{H\in\bigcap_{\ell\in\mathbb{N}}((VT_\ell^\eta)^{-1}(t_{E,\ell}))} [\![M]\!](H) \\ &= \inf_{\ell\in\mathbb{N}} \sum_{H\in(VT_\ell^\eta)^{-1}(t_{E,\ell})} [\![M]\!](H) \\ &= \inf_{\ell\in\mathbb{N}} (PT_\ell^\eta(M)(t_{E,\ell})).\end{aligned}$$

Let us prove point (2). On the one hand, $(PT^{\eta}_{\ell}(M)(t_{E,\ell}))_{\ell\in\mathbb{N}}$ is clearly a bounded below sequence. On the other hand, from (5.15) it is also monotone decreasing. Indeed, for all $\ell \in \mathbb{N}$:

$$\begin{aligned} PT^{\eta}_{\ell+1}(M)(t_{E,\ell+1}) &= \sum_{H\in(VT^{\eta}_{\ell+1})^{-1}(t_{E,\ell+1})} [\![M]\!](H) \\ &\leq \sum_{H\in(VT^{\eta}_{\ell})^{-1}(t_{E,\ell})} [\![M]\!](H) = PT^{\eta}_{\ell}(M)(t_{E,\ell}). \end{aligned}$$

Thus, $\lim_{\ell\to\infty}(PT^{\eta}_{\ell}(M)(t_{E,\ell}))_{\ell\in\mathbb{N}} = \inf_{\ell\in\mathbb{N}}(PT^{\eta}_{\ell}(M)(t_{E,\ell})) = [\![M]\!](E)$, and point (2) follows by definition of limit. □

Lemma 133. *Let $M, N \in \Lambda^{\emptyset}_{\oplus}$. If $M =_{\text{cxt}} N$ then $[\![M]\!](E) = [\![N]\!](E)$, for all $E \in \Lambda^{\emptyset}_{\oplus}/=_{\text{cxt}}$.*

Proof. Suppose toward contradiction that $[\![M]\!](E) \neq [\![N]\!](E)$ and consider $\epsilon > 0$ such that $2\epsilon \leq |[\![M]\!](E) - [\![N]\!](E)|$. By Lemma 132.(2) there exist $\ell \in \mathbb{N}$ such that:

$$[\![M]\!](E) \approx_{\epsilon} PT^{\eta}_{\ell}(M)(t_{E,\ell}) \qquad\qquad [\![N]\!](E) \approx_{\epsilon} PT^{\eta}_{\ell}(N)(t_{E,\ell}).$$

By Theorem 129, from $M =_{\text{cxt}} N$ we obtain $M =_{\text{PT}^{\eta}} N$, and hence $PT^{\eta}_{\ell}(M) = PT^{\eta}_{\ell}(N)$. By Fact 130, $[\![M]\!](E) \approx_{2\epsilon} [\![N]\!](E)$, i.e. $|[\![M]\!](E) - [\![N]\!](E)| < 2\epsilon$. A contradiction. □

Remark 17. Observe that the statement of Lemma 133 may fail when $\Lambda_{\oplus}$ is endowed with a different operational semantics than head reduction. As an example, recall the terms $M \triangleq \lambda xy.(x \oplus y)$ and $N \triangleq (\lambda xy.x) \oplus (\lambda xy.y)$ discussed in the introduction of this chapter (see (5.1)). In the lazy cbn, M and N are context equivalent [27]. Moreover, M is a value for lazy cbn, while N reduces with equal probability $\frac{1}{2}$ to $\mathbf{T} = \lambda xy.x$ and $\mathbf{F} = \lambda xy.y$. However, M, $\mathbf{T}$ and $\mathbf{F}$ are pairwise context inequivalent since, by setting $\mathcal{C} = [\cdot]\mathbf{I}\boldsymbol{\Omega}$, we have that $\mathcal{C}[M]$, $\mathcal{C}[\mathbf{T}]$, and $\mathcal{C}[\mathbf{F}]$ converge with probability $\frac{1}{2}$, 1, and 0, respectively. Therefore, by setting E as the lazy cbn context equivalent class containing M, we have $[\![M]\!](E) = 1$, while $[\![N]\!](E) = 0$.

We can now state and prove the fundamental result of this chapter:

Theorem 134 (Full abstraction)**.** *For all $M, N \in \Lambda_{\oplus}$:*

$$M =_{\text{cxt}} N \Leftrightarrow M \sim N.$$

Proof. The right-to-left direction is Theorem 126.(2). Concerning the converse, we first consider the case of closed terms. So, let $M, N \in \Lambda^{\emptyset}_{\oplus}$ be such that $M =_{\text{cxt}} N$. We prove that there exists probabilistic applicative bisimulation $\mathcal{R}$ containing $=_{\text{cxt}}$. We define $\mathcal{R}$ as follows:

$$\{(P,Q) \in \Lambda^{\emptyset}_{\oplus} \times \Lambda^{\emptyset}_{\oplus} \mid P =_{\text{cxt}} Q\} \cup \{(\nu x.H, \nu x.H') \in \widetilde{\text{HNF}} \times \widetilde{\text{HNF}} \mid \lambda x.H =_{\text{cxt}} \lambda x.H'\}.$$

Let us prove that $\mathcal{R}$ is a probabilistic applicative bisimulation. Since $=_{\text{cxt}}$ is an equivalence relation, then $\mathcal{R}$ is. Now, let $(\nu x.H, \nu x.H'), (P,Q) \in \mathcal{R}$, $E \in (\Lambda^{\emptyset}_{\oplus} \cup \widetilde{\text{HNF}})/\mathcal{R}$, and let $l \in \Lambda^{\emptyset}_{\oplus} \cup \{\tau\}$. We have to show that:

(1) $\mathcal{P}_{\oplus}(P, l, E) = \mathcal{P}_{\oplus}(Q, l, E)$,

(2) $\mathcal{P}_{\oplus}(\nu x.H, l, E) = \mathcal{P}_{\oplus}(\nu x.H', l, E)$.

Let us prove point (1). If $l \in \Lambda^{\emptyset}_{\oplus}$ then $\mathcal{P}_{\oplus}(P, L, E) = 0 = \mathcal{P}_{\oplus}(Q, L, E)$. If $l = \tau$ we define $\widehat{E} \triangleq \{\lambda x.H \in \text{HNF}^{\emptyset} \mid \nu x.H \in E\} \cup \{P' \in \Lambda^{\emptyset}_{\oplus} \mid P' \in E\}$. Then, by definition:

$$\mathcal{P}_{\oplus}(P, \tau, E) = [\![P]\!](\widehat{E}) \qquad \mathcal{P}_{\oplus}(Q, \tau, E) = [\![Q]\!](\widehat{E}).$$

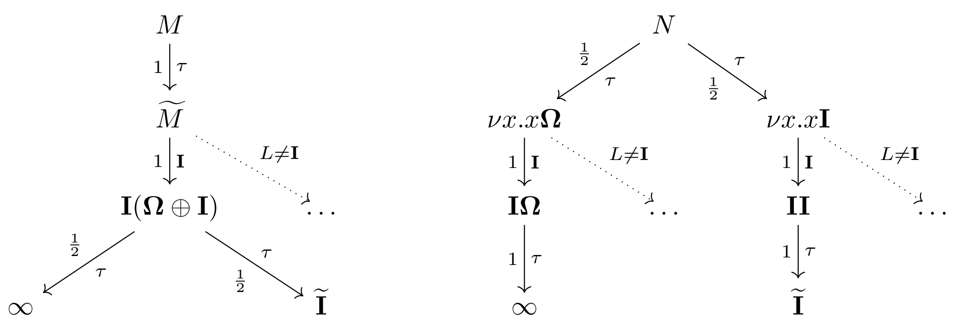


Figure 5.5: Markov chain for $M = \lambda x.x(\mathbf{\Omega} \oplus \mathbf{I})$ and $N = \lambda x.(x\mathbf{\Omega} \oplus x\mathbf{I})$.

Since $(P, Q) \in \mathcal{R}$ and $E \in (\Lambda^{\emptyset}_{\oplus} \cup \widetilde{\mathrm{HNF}})/\mathcal{R}$, it holds that $P =_{\mathrm{cxt}} Q$ and $\widehat{E} \in \Lambda^{\emptyset}_{\oplus}/_{=_{\mathrm{cxt}}}$. By applying Lemma 133 we have $[\![P]\!](\widehat{E}) = [\![Q]\!](\widehat{E})$, and hence $\mathcal{P}_{\oplus}(P, \tau, E) = \mathcal{P}_{\oplus}(Q, \tau, E)$. Let us now prove point (2). If $l = \tau$ then $P_{\oplus}(\nu x.H, \tau, E) = 0 = P_{\oplus}(\nu x.H', \tau, E)$. Otherwise, let $l = L \in \Lambda^{\emptyset}_{\oplus}$. Since $=_{\mathrm{cxt}} = \leq_{\mathrm{cxt}} \cap (\leq_{\mathrm{cxt}})^{op}$, by Lemma 117.(3) we have that $\lambda x.H =_{\mathrm{cxt}} \lambda x.H'$ implies $(\lambda x.H)L =_{\mathrm{cxt}} (\lambda x.H')L$. From Proposition 97.(2) and Proposition 100 we have:

$$H[L/x] =_{\mathrm{cxt}} (\lambda x.H)L =_{\mathrm{cxt}} (\lambda x.H')L =_{\mathrm{cxt}} H'[L/x].$$

Therefore, $H[L/x] \in E$ if and only if $H'[L/x] \in E$, and hence $\mathcal{P}_{\oplus}(\nu x.H, L, E) = \mathcal{P}_{\oplus}(\nu x.H', L, E)$.

Now, let $M, N \in \Lambda^{\{x_1,\ldots,x_n\}}_{\oplus}$ be such that $M =_{\mathrm{cxt}} N$. Since $=_{\mathrm{cxt}} = \leq_{\mathrm{cxt}} \cap (\leq_{\mathrm{cxt}})^{op}$, by repeatedly applying Lemma 117.(1) and Lemma 119.(1), $\lambda x_1 \ldots x_n.M =_{\mathrm{cxt}} \lambda x_1 \ldots x_n.N$. Since these terms are closed, we obtain $\lambda x_1 \ldots x_n.M \sim \lambda x_1 \ldots x_n.N$. Finally, from Definition 79 we conclude $M \sim N$. □

5.4.3 PAS is not complete

Theorem 134 establishes a precise correspondence between PAB and context equivalence. But what about PAS and context preorder? The Soundness Theorem (Theorem 126.(1)) states that the former implies the latter, so that it is natural to wonder whether the converse holds as well. Surprisingly enough, as in the case of the lazy reduction strategies (see [27] and [22]), the answer is negative.

A counterexample to PAS completeness, analogous to the one in [22] for cbv, is given by:

$$M \triangleq \lambda x.x(\mathbf{\Omega} \oplus \mathbf{I}) \qquad\qquad N \triangleq \lambda x.(x\mathbf{\Omega} \oplus x\mathbf{I}). \tag{5.16}$$

whose Markov chain is sketched in Figure 5.5. First, observe that M and N are incomparable with respect to PAS:

Lemma 135. *Neither $M \precsim N$ nor $N \precsim M$ hold.*

Proof. Let $M \precsim N$. Then, we have $\mathcal{P}_{\oplus}(M, \tau, \widetilde{M}) \leq \mathcal{P}_{\oplus}(N, \tau, \precsim(\widetilde{M}))$, so that $\nu x.x\mathbf{\Omega} \in \precsim(\widetilde{M})$, and $\widetilde{M} \precsim \nu x.x\mathbf{\Omega}$. Hence, $\mathcal{P}_{\oplus}(\widetilde{M}, \mathbf{I}, \mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I})) \leq \mathcal{P}_{\oplus}(\nu x.x\mathbf{\Omega}, \mathbf{I}, \precsim(\mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I})))$. This means that

$\mathbf{I\Omega} \in \precsim(\mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I}))$, so that $\mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I}) \precsim \mathbf{I\Omega}$. So $\frac{1}{2} = \mathcal{P}_{\oplus}(\mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I}), \tau, \widetilde{\mathbf{I}}) \leq \mathcal{P}_{\oplus}(\mathbf{I\Omega}, \tau, \precsim(\widetilde{\mathbf{I}})) = 0$. A contradiction.

Now, suppose $N \precsim M$. Then $\mathcal{P}_{\oplus}(N, \tau, \nu x.x\mathbf{I}) \leq \mathcal{P}_{\oplus}(M, \tau, \precsim(\nu x.x\mathbf{I}))$, so that $\widetilde{M} \in \precsim(\nu x.x\mathbf{I})$, and $\nu x.x\mathbf{I} \precsim \widetilde{M}$. Hence, $\mathcal{P}_{\oplus}(\nu x.x\mathbf{I}, \mathbf{I}, \mathbf{II}) \leq \mathcal{P}_{\oplus}(\widetilde{M}, \mathbf{I}, \precsim(\mathbf{II}))$. This means that $\mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I}) \in \precsim(\mathbf{II})$, so that $\mathbf{II} \precsim \mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I})$. Therefore, $1 = \mathcal{P}_{\oplus}(\mathbf{II}, \tau, \widetilde{\mathbf{I}}) \leq \mathcal{P}_{\oplus}(\mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I}), \tau, \precsim(\widetilde{\mathbf{I}})) = \frac{1}{2}$. A contradiction. □

However, the two terms can be compared through the context preorder relation. First, we need some technical lemmas.

Lemma 136. *For all $M \in \Lambda_{\oplus}$, $[\![M[\mathbf{\Omega}/x]]\!] \leq_{\mathfrak{D}} [\![M[\mathbf{I}/x]]\!]$.*

Proof. Let us consider the context $(\lambda x.M)[\cdot] \in \mathsf{C}\Lambda_{\oplus}$. Since $[\![\mathbf{\Omega}]\!] \leq_{\mathfrak{D}} [\![\mathbf{I}]\!]$, by Lemma 99.(1) we obtain $[\![(\lambda x.M)\mathbf{\Omega}]\!] \leq_{\mathfrak{D}} [\![(\lambda x.M)\mathbf{I}]\!]$. From Corollary 116, we conclude $[\![M[\mathbf{\Omega}/x]]\!] \leq_{\mathfrak{D}} [\![M[\mathbf{I}/x]]\!]$. □

Lemma 137. *For all $M \in \Lambda_{\oplus}$, $\sum[\![M[(\mathbf{\Omega} \oplus \mathbf{I})/x]]\!] \leq \frac{1}{2} \cdot \sum[\![M[\mathbf{\Omega}/x]]\!] + \frac{1}{2} \cdot \sum[\![M[\mathbf{I}/x]]\!]$.*

Proof. By Theorem 115 it is enough to prove the following inequation for all $n \in \mathbb{N}$:

$$\sum_{H \in \mathrm{HNF}} \mathcal{H}^n(M[(\mathbf{\Omega} \oplus \mathbf{I})/x], H) \leq \sum_{H \in \mathrm{HNF}} \frac{1}{2} \cdot \mathcal{H}^\infty(M[\mathbf{\Omega}/x], H) + \frac{1}{2} \cdot \mathcal{H}^\infty(M[\mathbf{I}/x], H). \qquad (5.17)$$

The proof is by induction on $(n, |M|)$, where $n \in \mathbb{N}$ and $|M|$ is the size of M, i.e. the number of nodes in the syntax tree of M. We have several cases:

- If $M = \lambda x.M'$ then, by using the induction hypothesis and Lemma 109.(2):

$$\begin{aligned}
\sum_{H \in \mathrm{HNF}} \mathcal{H}^n(M[(\mathbf{\Omega} \oplus \mathbf{I})/x], H) &= \sum_{\lambda x.H \in \mathrm{HNF}} \mathcal{H}^n(\lambda x.(M'[(\mathbf{\Omega} \oplus \mathbf{I})/x]), \lambda x.H) \\
&= \sum_{H \in \mathrm{HNF}} \mathcal{H}^n(M'[(\mathbf{\Omega} \oplus \mathbf{I})/x], H) \\
&\leq \sum_{H \in \mathrm{HNF}} \frac{1}{2} \cdot \mathcal{H}^\infty(M'[\mathbf{\Omega}/x], H) + \frac{1}{2} \cdot \mathcal{H}^\infty(M'[\mathbf{I}/x], H) \\
&= \sum_{H \in \mathrm{HNF}} \frac{1}{2} \cdot \mathcal{H}^\infty(M[\mathbf{\Omega}/x], H) + \frac{1}{2} \cdot \mathcal{H}^\infty(M[\mathbf{I}/x], H).
\end{aligned}$$

- Suppose now that M is a head normal form. From the previous case we can assume w.l.o.g. that M is a neutral term of the form $y\vec{P}$, where $\vec{P} = P_1 \dots P_m$ for some $m \in \mathbb{N}$ and $P_1, \dots, P_m \in \Lambda_{\oplus}$. If $y \neq x$ then $y\vec{P}[(\mathbf{\Omega} \oplus \mathbf{I})/x]$, $y\vec{P}[\mathbf{\Omega}/x]$, and $y\vec{P}[\mathbf{I}/x]$ are head normal forms, and the inequation in (5.17) is straightforward. Otherwise, $y = x$. If $n \geq 2$ then, by using the induction hypothesis, Lemma 109.(1), and Lemma 136, we have:

$$\begin{aligned}
\sum_{H \in \mathrm{HNF}} \mathcal{H}^n(M[(\mathbf{\Omega} \oplus \mathbf{I})/x], H) &= \sum_{H \in \mathrm{HNF}} \mathcal{H}^n((\mathbf{\Omega} \oplus \mathbf{I})\vec{P}[(\mathbf{\Omega} \oplus \mathbf{I})/x], H) \\
&= \sum_{H \in \mathrm{HNF}} \frac{1}{2} \cdot \mathcal{H}^{n-1}(\mathbf{\Omega}\vec{P}[(\mathbf{\Omega} \oplus \mathbf{I})/x], H) \\
&\quad + \frac{1}{2} \cdot \mathcal{H}^{n-1}(\mathbf{I}\vec{P}[(\mathbf{\Omega} \oplus \mathbf{I})/x], H) \\
&= \frac{1}{2} \cdot \sum_{H \in \mathrm{HNF}} \mathcal{H}^{n-2}(\vec{P}[(\mathbf{\Omega} \oplus \mathbf{I})/x], H)
\end{aligned}$$

$$
\begin{aligned}
&\le \frac{1}{2}\cdot \sum_{H\in \mathrm{HNF}} \frac{1}{2}\cdot \mathcal{H}^{\infty}(\vec{P}[\mathbf{\Omega}/x], H) + \frac{1}{2}\cdot \mathcal{H}^{\infty}(\vec{P}[\mathbf{I}/x], H)\\
&\le \frac{1}{2}\cdot \sum_{H\in \mathrm{HNF}} \frac{1}{2}\cdot \mathcal{H}^{\infty}(\vec{P}[\mathbf{I}/x], H) + \frac{1}{2}\cdot \mathcal{H}^{\infty}(\vec{P}[\mathbf{I}/x], H)\\
&= \sum_{H\in \mathrm{HNF}} \frac{1}{2}\cdot \mathcal{H}^{\infty}(\vec{P}[\mathbf{I}/x], H)\\
&= \sum_{H\in \mathrm{HNF}} \frac{1}{2}\cdot \mathcal{H}^{\infty}(\mathbf{\Omega}\vec{P}[\mathbf{\Omega}/x], H) + \frac{1}{2}\cdot \mathcal{H}^{\infty}(\mathbf{I}\vec{P}[\mathbf{I}/x], H)\\
&= \sum_{H\in \mathrm{HNF}} \frac{1}{2}\cdot \mathcal{H}^{\infty}(M[\mathbf{\Omega}/x], H) + \frac{1}{2}\cdot \mathcal{H}^{\infty}(M[\mathbf{I}/x], H).
\end{aligned}
$$

If $n < 2$ then $\mathcal{H}^n(M[(\mathbf{\Omega}\oplus\mathbf{I})/x]) = 0$.

- Last, suppose that M is not a head normal form. By using the induction hypothesis, Lemma 109.(1) and Lemma 109.(3), we have:

$$
\begin{aligned}
&\sum_{H\in \mathrm{HNF}} \mathcal{H}^n(M[(\mathbf{\Omega}\oplus\mathbf{I})/x], H) =\\
&= \sum_{H\in \mathrm{HNF}} \sum_{l+l'=n} \sum_{H'\in \mathrm{HNF}} \mathcal{H}^{l}(M, H')\cdot \mathcal{H}^{l'}(H'[(\mathbf{\Omega}\oplus\mathbf{I})/x], H)\\
&= \sum_{l+l'=n} \sum_{H'\in \mathrm{HNF}} \mathcal{H}^{l}(M, H')\cdot \left(\sum_{H\in \mathrm{HNF}} \mathcal{H}^{l'}(H'[(\mathbf{\Omega}\oplus\mathbf{I})/x], H)\right)\\
&= \sum_{\substack{l+l'=n\\ l'<n}} \sum_{H'\in \mathrm{HNF}} \mathcal{H}^{l}(M, H')\cdot \left(\sum_{H\in \mathrm{HNF}} \mathcal{H}^{l'}(H'[(\mathbf{\Omega}\oplus\mathbf{I})/x], H)\right)\\
&\le \sum_{H'\in \mathrm{HNF}} \mathcal{H}^{\infty}(M, H')\cdot \left(\sum_{H\in \mathrm{HNF}} \frac{1}{2}\cdot \mathcal{H}^{\infty}(H'[\mathbf{\Omega}/x], H) + \frac{1}{2}\cdot \mathcal{H}^{\infty}(H'[\mathbf{I}/x], H)\right)\\
&= \frac{1}{2}\cdot \sum_{H\in \mathrm{HNF}} \mathcal{H}^{\infty}(M[\mathbf{\Omega}/x], H) + \frac{1}{2}\cdot \sum_{H\in \mathrm{HNF}} \mathcal{H}^{\infty}(M[\mathbf{I}/x], H).
\end{aligned}
$$

This concludes the proof. □

Lemma 138. *It holds that $M \le_{\mathrm{cxt}} N$.*

Proof. By Lemma 119 it is enough to show that $M \le_{\mathrm{app}} N$. Since $M, N \in \Lambda_{\oplus}^{\emptyset}$, this amounts to check that, for all $n \in \mathbb{N}$ and for all $L_1, \ldots, L_n \in \Lambda_{\oplus}^{\emptyset}$, it holds that $\sum[\![ML_1 \ldots L_n]\!] \le \sum[\![NL_1 \ldots L_n]\!]$. The proof is by induction on $n \in \mathbb{N}$:

- If $n = 0$ then, by Proposition 97.(3) and Proposition 97.(4), we have: $\sum[\![M]\!] = 1 = \frac{1}{2}\cdot\sum[\![x\mathbf{\Omega}]\!] + \frac{1}{2}\cdot\sum[\![x\mathbf{I}]\!] = \sum[\![x\mathbf{\Omega}\oplus x\mathbf{I}]\!] = \sum[\![N]\!]$.
- Suppose $n = 1$. Then:

$$
\begin{aligned}
\sum[\![ML]\!] &= \sum[\![(\lambda x.x(\mathbf{\Omega}\oplus\mathbf{I}))L]\!]\\
&= \sum[\![L(\mathbf{\Omega}\oplus\mathbf{I})]\!] && \text{Prop. 97.(2)}
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{\lambda x.H \in \operatorname{supp}(\llbracket L \rrbracket)} \llbracket L \rrbracket(\lambda x.H) \cdot \sum \llbracket H[\mathbf{\Omega} \oplus \mathbf{I}/x] \rrbracket && \text{Prop. 97.(1)}\\
&\leq \frac{1}{2} \cdot \sum_{\lambda x.H \in \operatorname{supp}(\llbracket L \rrbracket)} \llbracket L \rrbracket(\lambda x.H) \cdot \sum \llbracket H[\mathbf{\Omega}/x] \rrbracket\\
&\quad + \frac{1}{2} \cdot \sum_{\lambda x.H \in \operatorname{supp}(\llbracket L \rrbracket)} \llbracket L \rrbracket(\lambda x.H) \cdot \sum \llbracket H[\mathbf{I}/x] \rrbracket && \text{Lem. 137}\\
&= \frac{1}{2} \cdot \sum \llbracket L\mathbf{\Omega} \rrbracket + \frac{1}{2} \cdot \sum \llbracket L\mathbf{I} \rrbracket && \text{Prop. 97.(1)}\\
&= \frac{1}{2} \cdot \sum \llbracket (x\mathbf{\Omega})[L/x] \rrbracket + \frac{1}{2} \cdot \sum \llbracket (x\mathbf{I})[L/x] \rrbracket\\
&= \sum_{H \in \operatorname{supp}(\llbracket x\mathbf{\Omega} \rrbracket) \cup \operatorname{supp}(\llbracket x\mathbf{I} \rrbracket)} \frac{1}{2} \cdot \Big(\llbracket x\mathbf{\Omega} \rrbracket + \llbracket x\mathbf{I} \rrbracket\Big)(H) \cdot \sum \llbracket H[L/x] \rrbracket\\
&= \sum_{H \in \operatorname{supp}\llbracket x\mathbf{\Omega} \oplus x\mathbf{I} \rrbracket)} \llbracket x\mathbf{\Omega} \oplus x\mathbf{I} \rrbracket(H) \cdot \sum \llbracket H[L/x] \rrbracket && \text{Prop. 97.(4)}\\
&= \sum_{\lambda x.H \in \operatorname{supp}(\llbracket N \rrbracket)} \llbracket N \rrbracket(\lambda x.H) \cdot \sum \llbracket H[L/x] \rrbracket && \text{Prop. 97.(3)}\\
&= \sum \llbracket NL \rrbracket. && \text{Prop. 97.(1)}
\end{aligned}
$$

- Finally, suppose $n > 1$. We define:

$$
\begin{aligned}
P &\triangleq ML_1 \dots L_{n-1}\\
Q &\triangleq NL_1 \dots L_{n-1}\\
r &\triangleq \sum_{\lambda x.H \in \operatorname{supp}(\llbracket Q \rrbracket)} \llbracket Q \rrbracket(\lambda x.H) \cdot \sum \llbracket H[L/x] \rrbracket\\
r' &= \sum_{\lambda x.H \in \operatorname{supp}(\llbracket P \rrbracket)} \llbracket P \rrbracket(\lambda x.H) \cdot \sum \llbracket H[L/x] \rrbracket.
\end{aligned}
$$

Since by induction hypothesis $0 \leq \sum \llbracket Q \rrbracket - \sum \llbracket P \rrbracket$, $r - r'$ must be positive. By Proposition 97.(1) this quantity is $\sum \llbracket QL_n \rrbracket - \sum \llbracket PL_n \rrbracket$. Therefore, $\sum \llbracket PL_n \rrbracket \leq \sum \llbracket QL_n \rrbracket$. □

Summing up, we have:

Theorem 139. PAS *is not complete (hence fully abstract) with respect to context preorder.*

5.4.4 Recovering full abstraction for PAS: a conjecture

In [58] Larsen and Skou present a language of "tests" for labelled Markov chains. Intuitively, a test can be seen as an algorithm for performing an experiment on a program: during the execution of a test, one can observe the success and the failure of the experiment with a given probability. In [93] van Breugel et al. prove that Larsen and Skou's testing equivalence on labelled Markov chains coincides with probabilistic bisimilarity. Moreover, they extend the language of tests with the so-called "disjunctive" ones and show that the resulting equivalence characterizes probabilistic similarity.

Let us introduce the language of tests:

Definition 87 (Testing language)**.** Let $(\mathcal{S},\mathcal{L},\mathcal{P})$ be a labelled Markov chain. The *testing language* $\mathcal{T}_{(\mathcal{S},\mathcal{L},\mathcal{P})}$ is given by the grammar:

$$t := \omega \mid a.t \mid (t,t)$$

where $a \in \mathcal{L}$. The *extended testing language* $\mathcal{T}^{\vee}_{(\mathcal{S},\mathcal{L},\mathcal{P})}$ is given by the grammar:

$$t := \omega \mid a.t \mid (t,t) \mid t \vee t$$

where $a \in \mathcal{L}$ and $t \vee t$ is called *disjunctive test.*

Roughly, the term ω represents the test that does nothing but successfully terminate. The term $a.t$ performs the action a and, in case of success, it proceeds with the test t. The test (t,t') makes two copies of the current state, allows both tests t and t' on each copy, and records success in case both sub-tests succeed. Finally, the test $t \vee t'$ makes two copies of the current state, allows both tests t and t' on each copy, and records success in case at least one sub-test succeeds.

Formally, the probability of success is defined as follows:

Definition 88 (Success probability)**.** Let $(\mathcal{S},\mathcal{L},\mathcal{P})$ be a labelled Markov chain. For all $s \in \mathcal{S}$ and $t \in \mathcal{T}^{\vee}_{(\mathcal{S},\mathcal{L},\mathcal{P})}$, we define:

$$\mathrm{Pr}_{\omega}(s) \triangleq 1 \qquad \mathrm{Pr}_{a.t}(s) \triangleq \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') \cdot \mathrm{Pr}_t(s')$$

$$\mathrm{Pr}_{(t,t')}(s) \triangleq \mathrm{Pr}_t(s) \cdot \mathrm{Pr}_{t'}(s) \qquad \mathrm{Pr}_{t \vee t'}(s) \triangleq \mathrm{Pr}_t(s) + \mathrm{Pr}_{t'}(s) - \mathrm{Pr}_t(s) \cdot \mathrm{Pr}_{t'}(s).$$

The following theorem states that testing equivalence characterizes probabilistic (bi)similarity on labelled Markov chains:

Theorem 140 (Testing equivalence [93])**.** *Let $(\mathcal{S},\mathcal{L},\mathcal{P})$ be a labelled Markov chain and let $s, s' \in \mathcal{S}$:*

(1) $s \sim s'$ if and only if $\mathrm{Pr}_t(s) = \mathrm{Pr}_t(s')$, for every $t \in \mathcal{T}_{(\mathcal{S},\mathcal{L},\mathcal{P})}$;

(2) $s \precsim s'$ if and only if $\mathrm{Pr}_t(s) \leq \mathrm{Pr}_t(s')$, for every $t \in \mathcal{T}^{\vee}_{(\mathcal{S},\mathcal{L},\mathcal{P})}$.

Example 32. Consider for example the terms in (5.16). Since Lemma 135 states that $M \precsim N$ does not hold, by Theorem 140.(2) there exists a test $t \in \mathcal{T}^{\vee}_{(\Lambda^{\emptyset}_{\oplus} \uplus \widetilde{\mathrm{HNF}},\, \Lambda^{\emptyset}_{\oplus} \uplus \{\tau\},\, \mathcal{P}_{\oplus})}$ such that $\mathrm{Pr}_t(M) > \mathrm{Pr}_t(N)$. It suffices to set $t \triangleq \tau.(\mathbf{I}.\tau.\omega) \vee (\mathbf{I}.\tau.\omega)$. Indeed, on the one hand:

$$\begin{aligned}
\mathrm{Pr}_t(M) &= 2 \cdot \mathrm{Pr}_{\mathbf{I}.\tau.\omega}(\widetilde{M}) - \mathrm{Pr}_{\mathbf{I}.\tau.\omega}(\widetilde{M})^2 = 2 \cdot \mathrm{Pr}_{\tau.\omega}(\mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I})) - \mathrm{Pr}_{\tau.\omega}(\mathbf{I}(\mathbf{\Omega} \oplus \mathbf{I}))^2 \\
&= \mathrm{Pr}_{\omega}(\widetilde{\mathbf{I}}) - \left(\frac{1}{2} \cdot \mathrm{Pr}_{\omega}(\widetilde{\mathbf{I}})\right)^2 = \frac{3}{4}.
\end{aligned}$$

On the other hand:

$$\begin{aligned}
\mathrm{Pr}_t(N) &= \frac{1}{2} \cdot \mathrm{Pr}_{(\mathbf{I}.\tau.\omega)\vee(\mathbf{I}.\tau.\omega)}(\nu x.x\mathbf{\Omega}) + \frac{1}{2} \cdot \mathrm{Pr}_{(\mathbf{I}.\tau.\omega)\vee(\mathbf{I}.\tau.\omega)}(\nu x.x\mathbf{I}) \\
&= \left(\mathrm{Pr}_{\mathbf{I}.\tau.\omega}(\nu x.x\mathbf{\Omega}) - \frac{1}{2} \cdot \mathrm{Pr}_{\mathbf{I}.\tau.\omega}(\nu x.x\mathbf{\Omega})^2\right) + \left(\mathrm{Pr}_{\mathbf{I}.\tau.\omega}(\nu x.x\mathbf{I}) - \frac{1}{2} \cdot \mathrm{Pr}_{\mathbf{I}.\tau.\omega}(\nu x.x\mathbf{I})^2\right) \\
&= \left(\mathrm{Pr}_{\tau.\omega}(\mathbf{I\Omega}) - \frac{1}{2} \cdot \mathrm{Pr}_{\tau.\omega}(\mathbf{I\Omega})^2\right) + \left(\mathrm{Pr}_{\tau.\omega}(\mathbf{II}) - \frac{1}{2} \cdot \mathrm{Pr}_{\tau.\omega}(\mathbf{II})^2\right) \\
&= \mathrm{Pr}_{\omega}(\widetilde{\mathbf{I}}) - \frac{1}{2} \cdot \mathrm{Pr}_{\omega}(\widetilde{\mathbf{I}})^2 = \frac{1}{2}.
\end{aligned}$$

The perfect matching between the (extended) testing equivalence and probabilistic similarity sheds lights on the meaning of Theorem 139. Let us see how. By Theorem 140, proving that PAS is complete for the context preorder amounts to show that, for every $M, N \in \Lambda_\oplus$, each test $t \in \mathcal{T}^{\vee}_{(\mathcal{S},\mathcal{L},\mathcal{P})}$ satisfying $\mathrm{Pr}_t(M) < \mathrm{Pr}_t(N)$ can be converted to a context $\mathcal{C}_t$ such that $\sum[\![\mathcal{C}_t[M]]\!] < \sum[\![\mathcal{C}_t[N]]\!]$. This means that we require contexts in $\Lambda_\oplus$ to simulate somehow the quantitative behaviour of tests. However, anybody familiar with the historical developments of the full abstraction problem for PCF [15, 76] would immediately regard disjunctive tests as something that cannot easily be implemented by terms in $\Lambda_\oplus$. Indeed, by looking at Definition 88, the probability of success for disjunctive tests is closely related to the observational behaviour of the term $[M \parallel N] \mapsto L$, a variant of Plotkin's *parallel disjunction* [76]. Intuitively, the operational meaning of parallel disjunction can be described as follows: if either the evaluation of M *or* the evaluation of N terminates, then the behaviour of $[M \parallel N] \mapsto L$ is the same as the behaviour of L, otherwise this term does not terminate. Thus, in a probabilistic setting, $[M \parallel N] \mapsto L$ converges to L with a probability that is equal to the probability that either M or N converge. Adding parallel disjunction to the probabilistic λ-calculus requires a further rule in Definition 70 when introducing a big-step probabilistic operational semantics:

$$\frac{M \Downarrow \mathscr{D} \qquad N \Downarrow \mathscr{E} \qquad L \Downarrow \mathscr{F}}{[M \parallel N] \mapsto L \Downarrow (\sum \mathscr{D} + \sum \mathscr{E} - (\sum \mathscr{D} \cdot \sum \mathscr{E})) \cdot \mathscr{F}}\, s6$$

In [23] the language $\Lambda_{\oplus,or}$ obtained by adding to $\Lambda_\oplus$ the parallel disjunction operator has been studied in a call-by-value setting: it turns out that shifting from $\Lambda_\oplus$ to $\Lambda_{\oplus,or}$ is enough to restore full abstraction for PAS.

We conjecture that the same happens when the head reduction strategy is considered. Let us motivate this claim. Suppose $\Lambda_{\oplus,or}$ is endowed with a probabilistic operational semantics $\langle\!\langle \cdot \rangle\!\rangle$ based on the head reduction satisfying the following equations for compositionality:

$$\langle\!\langle H \rangle\!\rangle = H \tag{5.18}$$

$$\langle\!\langle \lambda x.M \rangle\!\rangle = \lambda x.\langle\!\langle M \rangle\!\rangle \tag{5.19}$$

$$\begin{aligned}\langle\!\langle MN \rangle\!\rangle = &\sum_{\lambda x.P \,\in\, \mathrm{supp}(\langle\!\langle M \rangle\!\rangle)} \langle\!\langle M \rangle\!\rangle(\lambda x.P) \cdot \langle\!\langle P[N/x] \rangle\!\rangle \\ &+ \sum_{H \,\in\, \mathrm{supp}(\langle\!\langle M \rangle\!\rangle) \,\cap\, \mathrm{NEUT}} \langle\!\langle M \rangle\!\rangle(H) \cdot HN \end{aligned} \tag{5.20}$$

$$\langle\!\langle M \oplus N \rangle\!\rangle = \frac{1}{2} \cdot \langle\!\langle M \rangle\!\rangle + \frac{1}{2} \cdot \langle\!\langle N \rangle\!\rangle \tag{5.21}$$

$$\langle\!\langle [M \parallel N] \mapsto L \rangle\!\rangle = \Big(\sum \langle\!\langle M \rangle\!\rangle + \sum \langle\!\langle N \rangle\!\rangle - \Big(\sum \langle\!\langle M \rangle\!\rangle \cdot \sum \langle\!\langle N \rangle\!\rangle\Big)\Big) \cdot \langle\!\langle L \rangle\!\rangle. \tag{5.22}$$

First, note that from the equation in (5.20) it follows that:

$$\langle\!\langle (\lambda x.M)N \rangle\!\rangle = \langle\!\langle M[N/x] \rangle\!\rangle. \tag{5.23}$$

Then, let us consider the context $\mathcal{C} \triangleq (\lambda x.x(\lambda y.[y \parallel y] \mapsto \mathbf{I}))[\cdot]$. On the one hand, we have:

$$\begin{aligned}\langle\!\langle \mathcal{C}[M] \rangle\!\rangle &= \langle\!\langle (\lambda x.x(\lambda y.[y \parallel y] \mapsto \mathbf{I}))M \rangle\!\rangle \\ &= \langle\!\langle M(\lambda y.[y \parallel y] \mapsto \mathbf{I}) \rangle\!\rangle && \text{by (5.23)} \\ &= \langle\!\langle (\lambda y.[y \parallel y] \mapsto \mathbf{I})(\mathbf{\Omega} \oplus \mathbf{I}) \rangle\!\rangle && \text{by (5.23)} \\ &= \langle\!\langle [(\mathbf{\Omega} \oplus \mathbf{I}) \parallel (\mathbf{\Omega} \oplus \mathbf{I})] \mapsto \mathbf{I} \rangle\!\rangle && \text{by (5.23)}\end{aligned}$$

$$= \left(2 \cdot \sum\langle\!\langle \mathbf{\Omega} \oplus \mathbf{I} \rangle\!\rangle - \left(\sum\langle\!\langle \mathbf{\Omega} \oplus \mathbf{I} \rangle\!\rangle\right)^2\right) \cdot \mathbf{I} \qquad \text{by (5.22)}$$

$$= \left(\sum\langle\!\langle \mathbf{I} \rangle\!\rangle - \left(\frac{1}{2} \cdot \sum\langle\!\langle \mathbf{I} \rangle\!\rangle\right)^2\right) \cdot \mathbf{I} = \frac{3}{4} \cdot \mathbf{I} \qquad \text{by (5.21)}$$

On the other hand, we have:

$$\begin{aligned}
\langle\!\langle \mathcal{C}[N] \rangle\!\rangle &= \langle\!\langle (\lambda x.x(\lambda y.[y \parallel y] \mapsto \mathbf{I}))N \rangle\!\rangle \\
&= \langle\!\langle N(\lambda y.[y \parallel y] \mapsto \mathbf{I}) \rangle\!\rangle && \text{by (5.23)} \\
&= \sum_{\lambda x.P \in \operatorname{supp}(\langle\!\langle N \rangle\!\rangle)} \langle\!\langle N \rangle\!\rangle(\lambda x.P) \cdot \langle\!\langle P[(\lambda y.[y \parallel y] \mapsto \mathbf{I})/x] \rangle\!\rangle && \text{by (5.20)} \\
&= \sum_{\lambda x.P \in \operatorname{supp}(\langle\!\langle N \rangle\!\rangle)} \frac{1}{2} \cdot \langle\!\langle \lambda x.x\mathbf{\Omega} \rangle\!\rangle(\lambda x.P) \cdot \langle\!\langle P[(\lambda y.[y \parallel y] \mapsto \mathbf{I})/x] \rangle\!\rangle \\
&\quad + \frac{1}{2} \cdot \langle\!\langle \lambda x.x\mathbf{I} \rangle\!\rangle(\lambda x.P) \cdot \langle\!\langle P[(\lambda y.[y \parallel y] \mapsto \mathbf{I})/x] \rangle\!\rangle && \text{by (5.21)} \\
&= \frac{1}{2} \cdot \langle\!\langle (\lambda y.[y \parallel y] \mapsto \mathbf{I})\mathbf{\Omega} \rangle\!\rangle + \frac{1}{2} \cdot \langle\!\langle (\lambda y.[y \parallel y] \mapsto \mathbf{I})\mathbf{I} \rangle\!\rangle \\
&= \frac{1}{2} \cdot \langle\!\langle [\mathbf{\Omega} \parallel \mathbf{\Omega}] \mapsto \mathbf{I} \rangle\!\rangle + \frac{1}{2} \cdot \langle\!\langle [\mathbf{I} \parallel \mathbf{I}] \mapsto \mathbf{I} \rangle\!\rangle && \text{by (5.23)} \\
&= \frac{1}{2} \cdot \left(2 \cdot \sum\langle\!\langle \mathbf{\Omega} \rangle\!\rangle - \left(\sum\langle\!\langle \mathbf{\Omega} \rangle\!\rangle\right)^2\right) \cdot \mathbf{I} + \frac{1}{2} \cdot \left(2 \cdot \sum\langle\!\langle \mathbf{I} \rangle\!\rangle - \left(\sum\langle\!\langle \mathbf{I} \rangle\!\rangle\right)^2\right) \cdot \mathbf{I} && \text{by (5.22)} \\
&= \frac{1}{2} \cdot \mathbf{I}
\end{aligned}$$

Therefore, we have $\sum\langle\!\langle \mathcal{C}[N] \rangle\!\rangle = \frac{1}{2} < \frac{3}{4} = \sum\langle\!\langle \mathcal{C}[M] \rangle\!\rangle$, so that $M \leq_{\text{cxt}} N$ does not hold in $\Lambda_{\oplus,or}$.

Chapter 6

Conclusion and future developments

In this thesis we investigated non-laziness in both implicit complexity and probabilistic λ-calculus. We started with an analysis of the computational and proof-theoretical properties of LEM, a system able to exponentially compress Mairson and Terui's mechanisms of linear weakening and contraction [64, 65], and we explored its potential applications. Then we introduced LAM, a system endowed with a weaker version of the additive rules, called *linear additives*. The presence of linear additives is harmless from a complexity-theoretic viewpoint, and no lazy reduction strategy is required to prevent exponential explosions in normalization, as opposed to what happens with the standard additives. Also, we considered a probabilistic formulation of STA [40], called $\mathsf{STA}_\oplus$, with a non-deterministic variant of linear additives. $\mathsf{STA}_\oplus$ is able to capture the probabilistic polynomial time functions as well as the classes PP and BPP. Last, we presented the untyped probabilistic λ-calculus $\Lambda_\oplus$ endowed with an operational semantics based on head spine reduction, a variant of the head reduction strategy giving rise to the same big-step semantics. We have proven that probabilistic applicative bisimilarity is fully abstract with respect to context equivalence, showing that "non-laziness" is crucial to recover a correspondence between bisimilarity and context equivalence in the call-by-name probabilistic λ-calculus.

We conclude by briefly discussing possible future directions, chapter by chapter.

Chapter 3.

- In Section 3.2.2 we conjectured that a version of the general separation property for the λ-calculus [18] holds in the linear setting (Conjecture 7), and we showed how this result would imply the existence of a duplicator for every finite set of closed terms in $\beta\eta$-normal form, so connecting linear duplication with the standard notion of separation:

  $$\textit{linear separation} \sim \textit{linear duplication}$$

  To motivate our conjecture, let us consider the Böhm Theorem [17], a special case of the separation result in [18]. This theorem states that, for every pair of closed $\beta\eta$-normal forms M, N and all closed terms P, Q, a closed λ-term F exists such that $FM \to^*_\beta P$ and $FN \to^*_\beta Q$. Is it the case that F can be taken linear whenever M, N, P, Q are? It seems that the answer is positive, as the construction of F is essentially obtained by combining two kinds of λ-terms: the *permutators* (with shape $\lambda x_1 \dots x_n x_{n+1}.x_{n+1} x_1 \dots x_n$) and the *selectors* (with shape $\lambda x_1 \dots x_n.x_i$). Permutators are clearly linear terms, while selectors could be linearly defined, following Definition 5, as terms with form:

  $$\lambda x_1 \dots x_n.(x_1 \mathbf{I} \overset{k_1}{\dots} \mathbf{I}) \dots (x_{i-1} \mathbf{I} \overset{k_{i-1}}{\dots} \mathbf{I})(x_{i+1} \mathbf{I} \overset{k_{i+1}}{\dots} \mathbf{I}) \dots (x_n \mathbf{I} \overset{k_n}{\dots} \mathbf{I}) x_i$$

where $\mathbf{I} = \lambda x.x$ and each $k_1, \ldots, k_n$ must be large enough. Intuitively, $x_j \mathbf{I} \overset{k_j}{\ldots} \mathbf{I}$ is able to erase by linear consumption all potential closed linear λ-terms replacing x_j, provided that their size is at most k_j.

- Other possible future directions are suggested by the applications of LEM discussed in Section 3.4. On the one hand, we presented an encoding of the boolean circuits not preserving their depth. Moving to unbounded fan-in proof nets for LEM would improve the correspondence, where the rules p, w, c and d in Figure 3.5 would be expressed by nodes and boxes, like in Linear Logic. Operations on them would compactly perform duplication and get rid of garbage, possibly improving [89, 73, 8]. On the other hand, we contributed to the problem of defining numeral systems in linear settings. In [63], Mackie has recently introduced linear variants of numeral systems. He shows that successor, addition, predecessor, and subtraction have representatives in the linear λ-calculus. We could not find how giving type in LEM to some of the terms of Mackie's numeral systems. We conjecture that, by merging Mackie's encoding and Scott numerals [24], numeral systems exist which LEM can give a type to. The cost would be to extend LEM with recursive types, following Roversi and Vercelli [79].

Chapter 4. We strongly believe that linear additives can have fruitful applications in the field of implicit complexity, especially when the goal is to capture non-deterministic or probabilistic (sub-)polynomial complexity classes. A reasonable question in this framework could be following: is it possible to define a weaker version of the additive disjunction $\oplus$, let us denote it $\vee$, based on the same principles of the linear additive conjunction $\wedge$? We tried to answer this question by extending LAM with the following natural deduction rules for $\vee$:

$$\frac{\Gamma \vdash M : A_i \quad \vdash W : A_{3-i}}{\Gamma \vdash \mathtt{inj}_i^W(M) : A_1 \vee A_2} \vee\mathrm{I} \qquad \frac{\Gamma \vdash M : A_1 \vee A_2 \quad x_1 : A_1 \vdash M_1 : C \quad x_2 : A_2 \vdash M_2 : C}{\Gamma \vdash \mathtt{case}_C\, M \;\mathtt{of}\; [\mathtt{inj}_1(x_1) \to N_1 \mid \mathtt{inj}_2(x_2) \to N_2] : C} \vee\mathrm{E}$$

where A_1, A_2 and C are closed types free from negative occurrences of $\forall$, and W is an extended value (see Definitions 40 and 41). Let us call $\mathsf{LAM}^\vee$ the resulting system. According to this extension, the reduction relation $\to$ in Definition 41 must be endowed with the rule below:

$$\mathtt{case}_C\, \mathtt{inj}_i^{W'}(W) \;\mathtt{of}\; [\mathtt{inj}_1(x_1) \to N_1 \mid \mathtt{inj}_2(x_2) \to N_2] \to N_i[W/x_i] \tag{6.1}$$

The intended meaning of the above rules becomes apparent as soon as we define a translation of $\mathsf{LAM}^\vee$ into IMLL_2. This translation is obtained by extending Definition 43 with the following cases:

$$\begin{aligned}
(A_1 \vee A_2)^\bullet &\triangleq \mathbf{B} \otimes (A_1^\bullet \otimes A_2^\bullet) \\
\mathtt{inj}_1^W(M)^\bullet &\triangleq \langle \mathtt{tt}, \langle M^\bullet, W^\bullet \rangle\rangle \\
\mathtt{inj}_2^W(M)^\bullet &\triangleq \langle \mathtt{ff}, \langle W^\bullet, M^\bullet \rangle\rangle \\
(\mathtt{case}_C\, M \;\mathtt{of}\; [\mathtt{inj}_1(x_1) \to N_1 \mid \mathtt{inj}_2(x_2) \to N_2])^\bullet &\triangleq \mathtt{let}\; M^\bullet \;\mathtt{be}\; y, y' \;\mathtt{in}\; (\mathtt{let}\; y' \;\mathtt{be}\; y_1, y_2 \;\mathtt{in} \\
&\qquad (\mathtt{if}\; y \;\mathtt{then}\; N_1^\bullet[y_1/x_1] \;\mathtt{else}\; N_2^\bullet[y_2/x_2]))
\end{aligned}$$

where $\mathbf{B}$ is the type of booleans with inhabitants tt and ff as in (3.2) of Section 3.1.3, and the if-then-else construct is as in (3.10) of Section 3.2.2, the latter containing an eraser $\mathtt{E}_{C^\bullet}$ of type $C^\bullet$. Assuming Lemma 49 still holds, one can easily check that the following is a reduction in IMLL_2:

$$(\mathtt{case}_C\, \mathtt{inj}_i^{W'}(W) \;\mathtt{of}\; [\mathtt{inj}_1(x_1) \to N_1 \mid \mathtt{inj}_2(x_2) \to N_2])^\bullet \to_\beta^* N_i^\bullet[W^\bullet/x_i]$$

which allows us to prove a simulation result relating $\mathsf{LAM}^{\vee}$ and IMLL_2 similar to Theorem 50.

Nonetheless, from the viewpoint of implicit complexity, the new connective $\vee$ suffers from the same drawbacks as $\oplus$. Indeed, in order to fully evaluate terms we are forced to introduce the standard conversion rule (we omit types for the sake of simplicity):

$$\begin{array}{c}\texttt{case}\,(\texttt{case}\,M\ \texttt{of}\ [\texttt{inj}_1(x_1) \to N_1 \mid \texttt{inj}_2(x_2) \to N_2])\ \texttt{of}\ [\texttt{inj}_1(y_1) \to P_1 \mid \texttt{inj}_2(y_2) \to P_2] \\ \downarrow \\ \texttt{case}\,M\ \texttt{of}\ [\texttt{inj}_1(x_1) \to (\texttt{case}\,N_1\ \texttt{of}\ [\texttt{inj}_1(y_1) \to P_1 \mid \texttt{inj}_2(y_2) \to P_2]) \mid \\ \texttt{inj}_2(x_2) \to (\texttt{case}\,N_2\ \texttt{of}\ [\texttt{inj}_1(y_1) \to P_1 \mid \texttt{inj}_2(y_2) \to P_2])]\end{array} \tag{6.2}$$

which causes an exponential blow up in normalization analogous to those in IMALL_2 (see Proposition 41). We recall that the exponential explosion in IMALL_2 is due to the presence of implicit contractions in the inference rule &I, as stressed in Figure 4.2(b). To overcome this drawback, we designed the linear additives, where contraction is expressed in a “linear” way by exploiting the mechanism of linear duplication of Theorem 10. By contrast, the rule $\vee$E has a hidden form of *cocontraction*, i.e. an implication of the form $C \otimes C \multimap C$, which can hardly be expressed by some linear mechanism in IMLL_2. In analogy with Chapter 3, a possible solution could be the introduction of “lazy” reduction rules that forbid the conversion in (6.2). The result we would like to achieve in $\mathsf{LAM}^{\vee}$ is to restore a linear time normalization for terms having a special kind of type and according to a specific “lazy” reduction strategy, quite like in the case of LEM with Theorem 29. This investigation is left to future work.

Chapter 5. Our full abstraction result completes the picture about fully abstract descriptions of the probabilistic head reduction context equivalence, finally adding a coinductive characterisation. To the best of our knowledge, this picture can be resumed by the equivalences of all the following items, for M and N probabilistic λ-terms:

- M and N are context equivalent,
- M and N have the same probabilistic Nakajima tree [60, 61],
- M and N have the same denotation in the reflexive arena $\mathcal{U}$ of the cartesian closed category of probabilistic concurrent game semantics [20],
- M and N have the same denotation in the reflexive object $\mathcal{D}^{\infty}$ of the cartesian closed category of probabilistic coherence spaces or of the $\mathbb{R}^{+}$-weighted relations [20, 61],
- M and N are applicatively bisimilar (this thesis),
- M and N are testing equivalent according to the testing language $\mathtt{T}_0$ (a consequence of [92], here Theorem 140.(1), and this thesis).

In the final part of the chapter we introduced a counterexample to the full abstraction problem for probabilistic applicative similarity and we conjectured that extending the calculus with Plotkin’s parallel disjunction $[M \parallel N] \mapsto L$ (see [76]), as previously done in the call-by-value setting [23], is enough to restore this property. We also motivated our conjecture by showing that endowing the extended calculus $\Lambda_{\oplus,or}$ with a probabilistic operational semantics that satisfies some reasonable equations for compositionality is enough to circumvent the counterexample. A possible future work could be to prove this conjecture in $\Lambda_{\oplus,or}$.

Bibliography

[1] Samson Abramsky. The lazy λ- calculus. 1990.

[2] Samson Abramsky and C-H Luke Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105(2):159–267, 1993.

[3] Peter Aczel. An introduction to inductive definitions. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 739–782. Elsevier, 1977.

[4] Sandra Alves, Maribel Fernández, Mário Florido, and Ian Mackie. The Power of Linear Functions. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, volume 4207 of *Lecture Notes in Computer Science*, pages 119–134. Springer, 2006.

[5] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[6] Andrea Asperti. Light affine logic. In *Proceedings. Thirteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No. 98CB36226)*, pages 300–308. IEEE, 1998.

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

[8] Clément Aubert. Sublogarithmic uniform boolean proof nets. In Jean-Yves Marion, editor, *Proceedings Second Workshop on Developments in Implicit Computational Complexity, Saarbrücken, DICE 2011, Germany, April 2nd and 3rd, 2011.*, volume 75 of *EPTCS*, pages 15–27, 2011.

[9] Patrick Baillot and Virgile Mogbil. Soft lambda-calculus: a language for polynomial time computation. In *International Conference on Foundations of Software Science and Computation Structures*, pages 27–41. Springer, 2004.

[10] Hendrik Pieter Barendregt. The lambda calculus: Its syntax and semantics. 1984. *Studies in Logic and the Foundations of Mathematics*, 1984.

[11] Henk Barendregt, Wil Dekkers, and Richard Statman. *Lambda calculus with types*. Cambridge University Press, 2013.

[12] K. E. Batcher. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), pages 307–314, New York, NY, USA, 1968. ACM.

[13] Spephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the polytime functions. *Computational complexity*, 2(2):97–110, 1992.

[14] Stephen J Bellantoni, Karl-Heinz Niggl, and Helmut Schwichtenberg. Higher type recursion, ramification and polynomial time. *Annals of Pure and Applied Logic*, 104(1-3):17–30, 2000.

[15] Gérard Berry and Pierre-Louis Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20(3):265–321, 1982.

[16] Bard Bloom. Can lcf be topped? flat lattice models of typed λ-calculus. *Information and Computation*, 87(1-2):264–301, 1990.

[17] Corrado Böhm. Alcune proprieta delle forme β-η-normali nel λ-k-calcolo. *Pubblicazioni dell'Istituto per le Applicazioni del Calcolo*, 696:19, 1968.

[18] Corrado Böhm, Mariangiola Dezani-Ciancaglini, P Peretti, and S Ronchi Della Rocca. A discrimination algorithm inside λ-β-calculus. *Theoretical Computer Science*, 8(3):271–291, 1979.

[19] Johannes Borgström, Ugo Dal Lago, Andrew D Gordon, and Marcin Szymczak. A lambda-calculus foundation for universal probabilistic programming. In *ACM SIGPLAN Notices*, volume 51, pages 33–46. ACM, 2016.

[20] Pierre Clairambault and Hugo Paquet. Fully abstract models of the probabilistic lambda-calculus. In Dan R. Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK*, volume 119 of *LIPIcs*, pages 16:1–16:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[21] Alan Cobham. The intrinsic computational difficulty of functions. 1965.

[22] Raphaëlle Crubillé and Ugo Dal Lago. On probabilistic applicative bisimulation and call-by-value λ-calculi. In *European Symposium on Programming Languages and Systems*, pages 209–228. Springer, 2014.

[23] Raphaëlle Crubillé, Ugo Dal Lago, Davide Sangiorgi, and Valeria Vignudelli. On applicative similarity, sequentiality, and full abstraction. In *Correct System Design*, pages 65–82. Springer, 2015.

[24] H. B. Curry and R. Feys. *Combinatory Logic, Volume I.* North-Holland, 1958. Second printing 1968.

[25] Haskell B Curry. Functionality in combinatory logic. *Proceedings of the National Academy of Sciences of the United States of America*, 20(11):584, 1934.

[26] Haskell Brooks Curry, Robert Feys, William Craig, J Roger Hindley, and Jonathan P Seldin. *Combinatory logic*, volume 1. North-Holland Amsterdam, 1958.

[27] Ugo Dal Lago, Davide Sangiorgi, and Michele Alberti. On coinductive equivalences for higher-order probabilistic functional programs. 49(1):297–308, 2014.

[28] Ugo Dal Lago and Paolo Parisen Toldin. A higher-order characterization of probabilistic polynomial time. *Information and Computation*, 241:114–141, 2015.

[29] Ugo Dal Lago and Margherita Zorzi. Probabilistic operational semantics for the lambda calculus. *RAIRO-Theoretical Informatics and Applications*, 46(3):413–450, 2012.

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

[31] Luca De Alfaro. *Formal verification of probabilistic systems.* Number 1601. Citeseer, 1997.

[32] Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Probabilistic λ-calculus and quantitative program analysis. *Journal of Logic and Computation*, 15(2):159–179, 2005.

[33] Alejandro Díaz-Caro and Gilles Dowek. Non determinism through type isomorphism. *arXiv preprint arXiv:1303.7334*, 2013.

[34] Thomas Ehrhard, Michele Pagani, and Christine Tasson. The Ccomputational Meaning of Probabilistic Coherence Spaces. In Martin Grohe, editor, *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science (LICS 2011)*, IEEE Computer Society Press, pages 87–96, 2011.

[35] Thomas Ehrhard, Michele Pagani, and Christine Tasson. The computational meaning of probabilistic coherence spaces. In *2011 IEEE 26th Annual Symposium on Logic in Computer Science*, pages 87–96. IEEE, 2011.

[36] Thomas Ehrhard, Michele Pagani, and Christine Tasson. Probabilistic Coherence Spaces are Fully Abstract for Probabilistic PCF. In P. Sewell, editor, *The 41th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL14, San Diego, USA*. ACM, 2014.

[37] Thomas Ehrhard, Michele Pagani, and Christine Tasson. Full abstraction for probabilistic PCF. *J. ACM*, 65(4):23:1–23:44, 2018.

[38] Marco Gaboardi and Simona Ronchi Della Rocca. A soft type assignment system for λ-calculus. In *International Workshop on Computer Science Logic*, pages 253–267. Springer, 2007.

[39] Marco Gaboardi, Jean-Yves Marion, and Simona Ronchi Della Rocca. Soft linear logic and polynomial complexity classes. *Electronic Notes in Theoretical Computer Science*, 205:67–87, 2008.

[40] Marco Gaboardi and Simona Ronchi Della Rocca. From light logics to type assignments: a case study. *Logic Journal of the IGPL*, 17(5):499–530, 2009.

[41] Gerhard Gentzen. Untersuchungen über das logische schließen. i. *Mathematische zeitschrift*, 39(1):176–210, 1935.

[42] Jean-Yves Girard. Linear Logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

[43] Jean-yves Girard. logic: its syntax and semantics. In *Advances in Linear Logic*. Citeseer, 1995.

[44] Jean-Yves Girard. Light linear logic. *Information and Computation*, 143(2):175–204, 1998.

[45] Jean-Yves Girard. Proof-nets: the parallel syntax for proof-theory. In *Logic and Algebra*, pages 97–124. Routledge, 2017.

[46] Jean-Yves Girard and Yves Lafont. Linear logic and lazy computation. In *International Joint Conference on Theory and Practice of Software Development*, pages 52–66. Springer, 1987.

[47] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[48] J Roger Hindley. Bck-combinators and linear λ-terms have types. *Theoretical Computer Science*, 64(1):97–105, 1989.

[49] Martin Hofmann. A mixed modal/linear lambda calculus with applications to bellantoni-cook safe recursion. In *International Workshop on Computer Science Logic*, pages 275–294. Springer, 1997.

[50] Ross Horne. The sub-additives: A proof theory for probabilistic choice extending linear logic. In *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[51] William A Howard. The formulae-as-types notion of construction. *To HB Curry: essays on combinatory logic, lambda calculus and formalism*, 44:479–490, 1980.

[52] Douglas J Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124(2):103–112, 1996.

[53] Simona Kasterovic and Michele Pagani. The discriminating power of the let-in operator in the lazy call-by-name probabilistic lambda-calculus. In *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[54] Jan Willem Klop. Combinatory reduction systems. 1980.

[55] Kenneth Kunen and Jerry Vaughan. *Handbook of set-theoretic topology*. Elsevier, 2014.

[56] Yves Lafont. Soft linear logic and polynomial time. *Theoretical Computer Science*, 318(1):163–180, 2004.

[57] J. Laird, G. Manzonetto, G. McCusker, and M. Pagani. Weighted relational models of typed lambda-calculi. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2013), 25-28 June 2013, New Orleans, USA, Proceedings*, pages 301–310, 2013.

[58] Kim G Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and computation*, 94(1):1–28, 1991.

[59] D Leivant. Predicative recurrence and computational complexity i: word recurrence and poly-time. feasible mathematics ii, clote and remmel, 1994.

[60] Thomas Leventis. Probabilistic böhm trees and probabilistic separation. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 649–658. ACM, 2018.

[61] Thomas Leventis and Michele Pagani. Strong adequacy and untyped full-abstraction for probabilistic coherence spaces. In *International Conference on Foundations of Software Science and Computation Structures*, pages 365–381. Springer, 2019.

[62] Patrick Lincoln and John Mitchell. Operational aspects of linear lambda calculus. In *[1992] Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 235–246. IEEE, 1992.

[63] Ian Mackie. Linear Numeral Systems. *Journal of Automated Reasoning*, Feb 2018.

[64] Harry G. Mairson. Linear Lambda Calculus and PTIME-completeness. *J. Funct. Program.*, 14(6):623–633, November 2004.

[65] Harry G. Mairson and Kazushige Terui. On the Computational Complexity of Cut-Elimination in Linear Logic. In Carlo Blundo and Cosimo Laneve, editors, *Theoretical Computer Science*, pages 23–36, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[66] Christopher D Manning, Christopher D Manning, and Hinrich Schütze. *Foundations of statistical natural language processing.* MIT press, 1999.

[67] Satoshi Matsuoka. Nondeterministic linear logic. *arXiv preprint cs/0410029*, 2004.

[68] Satoshi Matsuoka. Weak typed Bohm Theorem on IMLL. *Annals of Pure and Applied Logic*, 145(1):37–90, 2007.

[69] Satoshi Matsuoka. Strong typed b\" ohm theorem and functional completeness on the linear lambda calculus. *arXiv preprint arXiv:1505.01326*, 2015.

[70] François Maurel. Nondeterministic light logics and np-time. In *International Conference on Typed Lambda Calculi and Applications*, pages 241–255. Springer, 2003.

[71] Robin Milner. Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4(1):1–22, 1977.

[72] John Mitchell, Mark Mitchell, and Andre Scedrov. A linguistic characterization of bounded oracle computation and probabilistic polynomial time. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 725–733. IEEE, 1998.

[73] Virgile Mogbil and Vincent Rahli. Uniform circuits, & boolean proof nets. In Sergei N. Artëmov and Anil Nerode, editors, *Logical Foundations of Computer Science, International Symposium, LFCS 2007, New York, NY, USA, June 4-7, 2007, Proceedings*, volume 4514 of *Lecture Notes in Computer Science*, pages 401–421. Springer, 2007.

[74] Reiji Nakajima. Infinite normal forms for the λ-calculus. In C. Böhm, editor, *λ-Calculus and Computer Science Theory*, pages 62–82, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.

[75] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Elsevier, 2014.

[76] Gordon D. Plotkin. Lcf considered as a programming language. *Theoretical computer science*, 5(3):223–255, 1977.

[77] Simonetta Ronchi Della Rocca and Luca Roversi. Lambda calculus and intuitionistic linear logic. *Studia Logica*, 59(3), 1997.

[78] Luca Roversi. A P-Time Completeness Proof for Light Logics. In *Ninth Annual Conference of the EACSL (CSL'99)*, volume 1683 of *Lecture Notes in Computer Science*, pages 469 – 483, Madrid (Spain), September 1999. Springer-Verlag.

[79] Luca Roversi and Luca Vercelli. Safe Recursion on Notation into a Light Logic by Levels. In *Proceedings of the Workshop on Developments in Implicit Computational complexity (DICE 2010)*, volume 23 of *Electronic Proceedings in Theoretical Computer Science*, pages 63 – 77. On-line, March 2010.

[80] Davide Sangiorgi. On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 31(4):15, 2009.

[81] Davide Sangiorgi. *Introduction to bisimulation and coinduction.* Cambridge University Press, 2011.

[82] Aleksy Schubert. The complexity of β-reduction in low orders. In *International Conference on Typed Lambda Calculi and Applications*, pages 400–414. Springer, 2001.

[83] Thomas Seiller. Probabilistic complexity classes through semantics. *arXiv preprint arXiv:2002.00009*, 2020.

[84] Peter Sestoft. Demonstrating lambda calculus reduction. In *The essence of computation*, pages 420–435. Springer, 2002.

[85] Alex Simpson. Reduction in a linear lambda-calculus with applications to operational semantics. In *International Conference on Rewriting Techniques and Applications*, pages 219–234. Springer, 2005.

[86] Michael Sipser. *Introduction to the Theory of Computation.* Cengage Learning, 2012.

[87] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism.* Elsevier, 2006.

[88] Kazushige Terui. Light affine lambda calculus and polytime strong normalization. In *Proceedings 16th Annual IEEE Symposium on Logic in Computer Science*, pages 209–220. IEEE, 2001.

[89] Kazushige Terui. Proof nets and boolean circuits. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 182–191. IEEE Computer Society, 2004.

[90] Anne Sjerp Troelstra and Helmut Schwichtenberg. *Basic proof theory.* Number 43. Cambridge University Press, 2000.

[91] Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. a correction. *Proceedings of the London Mathematical Society*, 2(1):544–546, 1938.

[92] Franck van Breugel, Michael Mislove, Joel Ouaknine, and James Worrel. Domain theory, testing and simulation for labelled markov processes. *Theoretical Computer Science*, 333(1):171 – 197, 2005. Foundations of Software Science and Computation Structures.

[93] Franck Van Breugel, Michael Mislove, Joël Ouaknine, and James Worrell. Domain theory, testing and simulation for labelled markov processes. *Theoretical Computer Science*, 333(1-2):171–197, 2005.

[94] Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach.* Springer-Verlag, Berlin, Heidelberg, 1999.

[95] Yu Zhang. The computational slr: A logic for reasoning about computational indistinguishability. In *International Conference on Typed Lambda Calculi and Applications*, pages 401–415. Springer, 2009.