Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

On the Completeness of a Syntactically Linear Logic

Tesis de Licenciatura en Ciencias de la Computación

Carlos Miguel Soto

Director: Alejandro Díaz-Caro

Buenos Aires, 2024

SOBRE LA COMPLETITUD DE UNA LÓGICA SINTÁCTICAMENTE LINEAL

El cálculo $\mathcal{L}^{\mathcal{S}}$, una extensión del lenguaje de pruebas de la lógica lineal relacionado con la computación cuántica, ha sido dotado de una semántica concreta en [DCM23]. En este artículo, demostramos resultados de completitud para esta semántica con respecto a la equivalencia computacional. Introducimos una semántica operacional para $\mathcal{L}^{\mathcal{S}}$ y demostramos su isomorfismo con la semántica categórica bajo ciertas condiciones. Además, mostramos que la categoría se puede restringir a semimódulos finitamente generados, los cuales poseen mejores propiedades.

Palabras Clave: Cuántica, Lógica Lineal, Semántica Categórica, Completitud, Semimódulos.

ON THE COMPLETENESS OF A SYNTACTICALLY LINEAR LOGIC

The $\mathcal{L}^{\mathcal{S}}$ calculus, an extension of the proof language of linear logic related to quantum computing, has been given a concrete semantics in [DCM23]. In this paper, we prove a full abstraction result for this semantics with respect to a notion of computational equivalence. We introduce an operational semantics for $\mathcal{L}^{\mathcal{S}}$ and prove its isomorphism to the categorical semantics under certain conditions. Additionally, we show that the category can be restricted to that of finitely generated semimodules, which possess better properties.

Keywords: Quantum, Linear Logic, Categorical Semantics, Completeness, Semimodules.

AGRADECIMIENTOS

Quiero agradecer a mis papás por haberme apoyado a hacer lo que me gusta, incluso cuando todavía no sabía qué era. Agradezco también a Ivo por su exasperante meticulosidad con los TPs, contagiosa valentía para rendir finales, y amistad.

Finalmente agradezco a Jano por su dirección y mentoría académica.

CONTENTS

1. PRELIMINARIES

1.1 Curry-Howard Correspondence

In [How80], the author presented what is now known as the Curry-Howard correspondence. This correspondence establishes a connection between certain proofs for positive implicational calculus and terms of a typed lambda calculus.

The propositions considered in [How80] were very simple: only propositional variables and implication were allowed. The proofs were similarly restricted, and were based on *sequents*. Sequents are a pair of hypotheses (also called the context) together with a conclusion, written $\Gamma \vdash \phi$ where Γ is a set of formulas (hypotheses) and ϕ is a single formula (the conclusion). Only four rules were allowed to recursively construct proofs:

- The axiom $\alpha \vdash \alpha$, that is if the only hypothesis is a formula, you can conclude that formula. (ax.)
- If $\Gamma, \alpha \vdash \beta$ then $\Gamma \vdash \alpha \to \beta$. That is, if β can be proven assuming α, then the implication $\alpha \to \beta$ can be proven as well. (impl.)
- Modus ponens. That is, if $\Gamma \vdash \alpha$ and $\Delta \vdash \alpha \to \beta$ then $\Gamma, \Delta \vdash \beta$. (mp.)
- Structural rules, which allow deduplicating, reordering, and adding unused formulas to the hypotheses. (struc.)

This leads to proofs that are structured as trees. For example, a proof for

$$P \to (Q \to (R \to P))$$

could be constructed as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{P \vdash P}\,\text{ax.}}{P, R \vdash P}\,\text{struc.}}{P \vdash R \to P}\,\text{impl.} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{R \to P \vdash R \to P}\,\text{ax.}}{Q, R \to P \vdash R \to P}\,\text{struc.}}{R \to P \vdash Q \to (R \to P)}\,\text{impl.}}{\vdash (R \to P) \to (Q \to (R \to P))}\,\text{impl.}}{\dfrac{P \vdash Q \to (R \to P)}{\vdash P \to (Q \to (R \to P))}\,\text{impl.}}\,\text{mp.}$$

The key insight in [How80] is that these proofs correspond to terms of a programming language. For example, the proof above corresponds to the term

$$\lambda x^P.(\lambda d^{R \to P}.\lambda y^Q.d)(\lambda z^R.x)$$

Where the superscripts indicate the type of the variable. Here, we are encoding instances of the impl. rule as lambda abstractions, instances of modus ponens as function application

and the ax. rule as simply using a variable when it is available in the context. The fact that the language allows unused variables implicitly encodes the structural rules.

However, the correspondence is even deeper. It was discovered that all instances of the modus ponens rule where the implication is directly constructed (i.e., the function being applied is a lambda abstraction, rather than a variable) could be eliminated, using a proof simplification transformation known as cut elimination. This technique, when translated to the language of lambda calculus, is exactly β-reduction. Using β-reduction, the above proof could be simplified to

$$\lambda x^P.\lambda y^Q.\lambda z^R.x$$

This theorem is important because it allows us to transfer normalization results from lambda calculus to proofs in logic, and allows us to study the behaviour of both proofs and programs modulo β-equivalence.

The Curry-Howard isomorphism has since been generalized in two ways. Firstly, the very weak positive implicational logic can be extended or restricted in various ways, obtaining different associated lambda-calculi. For example, the logic can be extended with a conjunction operator (yielding positive intuitionistic logic) which gives rise to a lambda-calculus with pairs and its neutral element "truth". Another example (which we will study in this paper) is the elimination of the structural rules, which gives rise to linear logic and its associated lambda calculus, where variables must be used exactly once. This logic was introduced by Girard in [Gir87].

Secondly, Lambek discovered in [Lam68] that the correspondence between terms and proofs could be extended with a third leg: morphisms in a category. A more extensive recount on these topics can be found in [LS86].

1.2 Curry-Howard-Lambek Correspondence

Definition 1.2.1. *Category. A category $\mathcal{C}$ is a set of objects $ob(\mathcal{C})$ together with a set of morphisms between said objects. The set of morphisms between two objects $A, B \in ob(\mathcal{C})$ is denoted $Hom(A, B)$. The morphisms must* compose. *That is, for any three objects A, B, C in the category, there is a composition operator $\circ : Hom(B, C) \times Hom(A, B) \to Hom(A, C)$. This operator must satisfy:*

- *Associativity: $f \circ (g \circ h) = (f \circ g) \circ h$*
- *Identity: For every object A there is an identity morphism $id_A \in Hom(A, A)$ such that $f \circ id_A = f = id_B \circ f$ for all $f \in Hom(A, B)$.*

The most common example of a category is the category of sets where every object is a set and $\mathrm{Hom}(A, B)$ is the set of all functions between A and B. Another example is the category of vector spaces whose objects are vector spaces and $\mathrm{Hom}(A, B)$ is the set of linear transformations between A and B.

The Lambek correspondence is a certain surjective embedding of proofs into the morphisms of a category. In [LS86], for example, they exemplify this correspondence in the context of positive intuitionistic logic. They considered a category where the objects are all types of the logic (which are formed by the base types and the type constructors $\to$ and $\wedge$) and the morphisms represent certain equivalence classes of terms. Terms cannot be directly mapped to morphisms because they do not exactly obey the associativity and identity laws of categories, but they do so up to β-equivalence. It was proven that the resulting category is a cartesian closed category, where the $\wedge$ type constructor acts as a categorical product (with truth as the unit) and the $\to$ constructor acts as the internal hom.

Definition 1.2.2. *Cartesian Closed Category. A cartesian closed category is a category where there is a tensor product $\times$, that takes two objects and produces another, with the following properties:*

1. *The $\times$ must be a cartesian product, that is, for any two objects A, B there should be two morphisms $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$, called the projections, and $A \times B$ must be universal with respect to those.*
2. *The $\times$ operator must be adjoint with Hom. That is, the objects $Hom(A, Hom(B, C))$ and $Hom(A \times B, C)$ must be naturally isomorphic.*
3. *There must exist an object, $\top$, that acts as a neutral element for $\times$.*

Different logics give rise to different types of category. For example, linear logic results in monoidal categories (see [BS04]). These are categories in which the operator $\times$ need not be a cartesian product, in which case it is denoted by $\otimes$.

1.3 Denotational Semantics

For studying the properties of a logic or a programming language, in many settings, it is often useful to have explicit mappings from the terms of the language into a chosen category, instead of the one given by the Curry-Howard-Lambek correspondence. For example, when studying linear logic as a quantum programming language, it is productive to work in a category of vector spaces as opposed to an abstract category based on the terms of the language because the laws of quantum mechanics are more naturally expressed in that setting. This can be done though a so called *denotational semantics*.

A denotational semantics is a mapping from the terms of a language into a chosen category that respects the structure of the language. More specifically, the mapping must be invariant under all reduction rules of the language, like cut elimination. A key difference over the mapping given by Curry-Howard-Lambek correspondence is that the morphisms in a denotational semantics must be recursively defined. That is, if a term is composed of subterms, then the morphism associated with the term must be the result of applying certain operation on the morphism associated with the subterms.

As explained previously, a denotational semantics cannot distinguish all terms, because associativity of term composition only holds up to suitable reduction rules. Thus, a very

important property of denotational semantics is *which terms they equate*. A denotational semantics that equates all terms, for example, is not very useful because it cannot be used to differentiate different programs with the same types. Ideally, we would like exactly the terms that "behave the same" to be equated. This property, we will see, is very closely related to the extent to which the mapping is surjective. That is, if for every morphism in the category there is a term that maps to it. We will formalize and study both of these properties for a proof system called $\mathcal{L}^{\mathcal{S}}$, defined in [DCD24].

1.4 Linear Logic

The proof system of study, $\mathcal{L}^{\mathcal{S}}$, is a proof system for *intuitionistic linear logic*. That is, it results from intuitionistic logic but with a removal of the structural rules. In intuitionistic logic there are two equivalent definitions for the introduction rule of the $\wedge$:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge_i \qquad \text{and} \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B}\ \wedge_i$$

The first one implies that if a context proves both A and B, then it also proves $A \wedge B$. The second one implies that if two contexts prove A and B respectively, then the union of the contexts proves $A \wedge B$. In linear logic, both rules are preserved but correspond to different connectives:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\ \&_i \qquad \text{and} \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ \otimes_i$$

Which come with their corresponding elimination rules

$$\frac{\Gamma \vdash A \,\&\, B \quad \Delta, A \vdash C}{\Gamma, \Delta \vdash C}\ \&_e^1 \qquad \frac{\Gamma \vdash A \,\&\, B \quad \Delta, B \vdash C}{\Gamma, \Delta \vdash C}\ \&_e^2$$

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

Notice that for the & connective, which is referred to as the *additive conjunction*, creating a proof of $A \,\&\, B$ requires using the whole context on both sides, but then eliminating a term of type $A \,\&\, B$ only requires using one of the sides. Conversely, when introducing a term of type $A \otimes B$ the context has to be partitioned in two, some of it going to the left side and some to the right side, but then eliminating a term of type $A \otimes B$ requires using both elements. This ensures that values can only be duplicated in a controlled way through special use of the connectives.

For example, the diagonal function term $\lambda x.\langle x, x\rangle$ that creates an additive pair of type $A \,\&\, A$ with an equal value on both sides is a valid term, but the expression $\lambda x.x \otimes x$ that creates a tensor pair of type $A \otimes A$ is not. Whereas the term $\lambda x.\lambda y.\langle x, y\rangle$ is not valid, while the term $\lambda x.\lambda y.x \otimes y$ is.

There is an analogous situation in linear algebra. The function $f : \mathbb{R} \to \mathbb{R}^2, f(x) = (x, x)$ is linear while the function $g : \mathbb{R} \to \mathbb{R} \otimes \mathbb{R}, g(x) = x \otimes x$ is not. And analogously, the function $f(x, y) = (x, y)$ is not *bilinear* in its two arguments, while the function $g(x, y) = x \otimes y$

is. This is a hint that we can encode terms in linear logic as multilinear morphisms in an appropriate linear category.

2. INTRODUCTION

2.1 Definitions

The aim of this thesis is to study the soundness and completeness of a denotational semantics for the $\mathcal{L}$-calculus defined in [DCD24]. To formalize this, let us start with some definitions:

Definition 2.1.1. *Soundness. Given an equivalence relation among the terms of the language* $R \subseteq \text{terms} \times \text{terms}$*, a denotational semantics that maps terms onto a category* $[\![.]\!] : \text{terms} \to \mathcal{C}$ *is said to be* sound *if* $t_1 R t_2$ *implies* $[\![t_1]\!] = [\![t_2]\!]$.

Definition 2.1.2. *Completeness. Given an equivalence relation* R *and denotational semantics* $[\![.]\!]$ *as above, the semantics is said to be* complete *with respect to the relation if* $[\![t_1]\!] = [\![t_2]\!]$ *implies* $t_1 R t_2$.

Definition 2.1.3. *Full Abstraction. A denotational semantics is said to be* fully abstract *with respect to an equivalence relation if it is both sound and complete.*

The two equivalence relations we consider between terms are

- The symmetric-transitive closure of the reduction rules of the language ($\longleftrightarrow^*$).
- Computational equivalence ($\equiv$), which requires correducibility ($\longleftrightarrow^*$) after replacing the terms into arbitrary $\mathtt{1}$ evaluation contexts. This is a weaker condition than $\longleftrightarrow^*$, since it does not require direct correducibility of the terms themselves. This notion will be formalized later.

2.2 Results

In [DCM23] it was proven that $[\![.]\!]$ is sound with respect to $\longleftrightarrow^*$, but it is easy to see that completeness fails for this equivalence relation. For this reason, the notion of computational equivalence was introduced, and it was proven that the given semantics are complete with respect to it. Thus, we set to investigate the soundness of the proposed semantics with respect to computational equivalence to ascertain its full abstraction.

We found that the semantics is not sound, but it can be made sound when it is restricted to a certain subset of the types of the language. Furthermore, we express the full abstraction for the semantics over these types as an isomorphism between the denotational and a newly defined operational semantics.

2.3 Organization

Chapters 1, 3, 4 and 6 present background material: Chapter 1 recalls the Curry-Howard-Lambek correspondence and denotational semantics; Chapter 3 collects standard results on semirings, semimodules and their tensor products; Chapter 4 reproduces the language $\mathcal{L}^{\mathcal{S}}$ and its reduction rules from [DCD24]; and Chapter 6 reproduces the denotational semantics defined in [DCM23].

The original contributions of this thesis are contained in Chapters 5, 7 and 8: the proof that weak and strong computational equivalence coincide (Chapter 5); the definition of the operational category and its semimodule structure (Chapter 7); soundness counterexample for the denotational semantics and isomorphism between the denotational and operational semantics for the well-behaved types (Chapter 8).

3. ALGEBRAIC STRUCTURES

3.1 Semirings and Semimodules

Throughout this thesis, $\mathcal{S}$ represents a fixed commutative semiring.

Definition 3.1.1 (Commutative Semiring)**.** *A commutative semiring is a set $\mathcal{S}$ equipped with two operations $+_{\mathcal{S}}$ and $\cdot_{\mathcal{S}}$ and two distinguished elements $0_{\mathcal{S}}, 1_{\mathcal{S}} \in \mathcal{S}$ such that:*

- $(\mathcal{S}, +_{\mathcal{S}})$ *forms a commutative monoid with identity* 0.
- $(\mathcal{S}, \cdot_{\mathcal{S}})$ *forms a commutative monoid with identity* 1.
- *For any* $a, b, c \in \mathcal{S}$, $a \cdot_{\mathcal{S}} (b +_{\mathcal{S}} c) = a \cdot_{\mathcal{S}} b +_{\mathcal{S}} a \cdot_{\mathcal{S}} c$ *(distributive property).*
- *For any* $s \in \mathcal{S}$, $s \cdot_{\mathcal{S}} 0_{\mathcal{S}} = 0_{\mathcal{S}}$.

The suffix $_{\mathcal{S}}$ can be omitted when clear from context.

The denotational semantics takes place over a subcategory of the category of semimodules of $\mathcal{S}$.

Definition 3.1.2 (Semimodule)**.** *A semimodule is a commutative monoid $(\mathcal{M}, +_{\mathcal{M}})$ with identity $0_{\mathcal{M}}$ together with a* scalar multiplication *$(s, \mu) \mapsto s\mu$ for any $s \in \mathcal{S}$ and $\mu \in \mathcal{M}$ that satisfy:*

- $(a \cdot b)\mu = a(b\mu)$
- $a(\mu +_{\mathcal{M}} \nu) = a\mu +_{\mathcal{M}} a\nu$
- $(a + b)\mu = a\mu +_{\mathcal{M}} b\mu$
- $1_{\mathcal{S}}\mu = \mu$
- $a0_{\mathcal{M}} = 0_{\mathcal{S}}\mu = 0_{\mathcal{M}}$

The suffix $_{\mathcal{M}}$ can be omitted when clear from context.

3.2 Hom functor

Definition 3.2.1 (Semimodule Morphism)**.** *A function $f : \mathcal{M} \to \mathcal{M}'$ between two semimodules is said to be a* morphism *if it satisfies*

- $sf(\mu) = f(s\mu)$, *and*
- $f(\mu + \nu) = f(\mu) + f(\nu)$

For any $\mu, \nu \in \mathcal{M}$ and $s \in \mathcal{S}$. We denote this $f \in Hom(\mathcal{M}, \mathcal{M}')$. $Hom(\mathcal{M}, \mathcal{M}')$ can be endowed with a semimodule structure with pointwise addition and multiplication by scalar.

The semimodules used in the denotational semantics are the free and finitely generated semimodules.

Definition 3.2.2 (Free Finitely Generated Semimodule (ffg))**.** *A semimodule $\mathcal{M}$ is said to be* free and finitely generated *if there exists a finite subset $B \subseteq \mathcal{M}$, a* base*, such that for any semimodule $\mathcal{N}$ and any function $f : B \to \mathcal{N}$ there exists a unique morphism $\Phi : \mathcal{M} \to \mathcal{N}$ such that $\Phi(b) = f(b)$ for all $b \in B$.*

Theorem 3.2.3. *Let M be any finite set. Then the semimodule $\mathcal{S}^M$ consisting of formal linear combinations of elements of M with arbitrary coefficients $(s_\mu)_{\mu \in M}$, i.e.*

$$\mathcal{S}^M = \left\{ \sum_{\mu \in M} s_\mu \cdot \mu \quad : \quad s_\mu \in \mathcal{S} \right\}$$

is an ffg with base M.

Proof. Let $\mathcal{N}$ be a semimodule, then every function $f : M \to \mathcal{N}$ can be extended to a morphism $\Phi : \mathcal{S}^M \to \mathcal{N}$ by

$$\Phi \left(\sum_{\mu \in M} s_\mu \cdot \mu \right) = \sum_{\mu \in M} s_\mu \cdot f(\mu)$$

This morphism is unique because M generates $\mathcal{S}^M$. □

Definition 3.2.4 (Dual base)**.** *For a ffg semimodule $\mathcal{M}$ with base $M \subseteq \mathcal{M}$, we define its dual base as the set*

$$M^* = \{\mu^* : \mu \in M\} \subseteq Hom(\mathcal{M}, \mathcal{S})$$

Where μ^* is the unique morphism that equals zero on all elements of M except for μ, where it equals 1*.*

Theorem 3.2.5 (Linear combination of base)**.** *Let $\mathcal{M}$ be a ffg semimodule with base M, then every element $x \in \mathcal{M}$ can be expressed as a linear combination of the elements of the base as follows*

$$x = \sum_{\mu \in M} \mu^*(x) \cdot \mu$$

Where $\mu^* \in M^*$ is the element of the dual base corresponding to μ.

Proof. Consider the function

$$f(x) = \sum_{\mu \in M} \mu^*(x) \cdot \mu$$

The function f is linear by construction. Let $\mu' \in M$ be any element of the base, then we have that

$$f(\mu') = \sum_{\mu \in M} \mu^*(\mu') \cdot \mu$$

Since $\mu^*(\mu')$ is, by definition, zero if $\mu \neq \mu'$ and one if $\mu = \mu'$, all terms of the summation cancel out except one, and we have

$$f(\mu') = 1 \cdot \mu'$$

Thus f is the identity on all elements of the base. Since f is linear and it coincides over a base with the identity (which is also linear), it follows from the unicity in the definition of base that f coincides with the identity on the entire semimodule, which is what we wanted to prove. □

Theorem 3.2.6 (Hom of a ffg)**.** *Let $\mathcal{M}$ and $\mathcal{N}$ be ffg semimodules with bases M and N respectively, then $Hom(\mathcal{M},\mathcal{N})$ is ffg with base*

$$B = \{b_{\mu\nu} : \mu \in M, \nu \in N\}$$

Where

$$b_{\mu\nu} : Hom(\mathcal{M},\mathcal{N}) \quad b_{\mu\nu}(x) = \mu^*(x) \cdot \nu$$

Proof. Consider any mapping $T : B \mapsto \mathcal{V}$ where B is the proposed base of $\mathrm{Hom}(\mathcal{M},\mathcal{N})$ and $\mathcal{V}$ is any semimodule. We can define a morphism $\Phi : \mathrm{Hom}(\mathcal{M},\mathcal{N}) \to \mathcal{V}$ by

$$\Phi(f) = \sum_{\mu \in M, \nu \in N} \nu^*(f(\mu)) \cdot T(b_{\mu\nu})$$

This is a morphism by construction, and since $\nu^*(b_{\mu'\nu'}(\mu))$ is 1 if $\nu = \nu'$ and $\mu = \mu'$ and 0 otherwise, the morphism Φ is an extension of T to the whole space.

The uniqueness of Φ follows from the fact that every $f \in \mathrm{Hom}(\mathcal{M},\mathcal{N})$ can be expressed as a linear combination of the elements of the base:

$$f(x) = \sum_{\mu \in M, \nu \in N} \nu^*(f(\mu)) \cdot b_{\mu\nu}$$

To verify this equality, let us evaluate both sides:

$$f(x) = \sum_{\mu \in M, \nu \in N} \nu^*(f(\mu)) \cdot b_{\mu\nu}(x) = \sum_{\mu \in M, \nu \in N} \nu^*(f(\mu)) \cdot \mu^*(x) \cdot \nu$$

Notice that the function

$$x \mapsto \sum_{\mu \in M, \nu \in N} \nu^*(f(\mu)) \cdot \mu^*(x) \cdot \nu$$

Coincides with $f(x)$ for all elements of the base of $\mathcal{M}$ due to 3.2.5, and is linear by construction. Therefore, it is exactly f. □

This means that the set of ffg semimodules forms a subcategory of the category of semimodules that is closed under the Hom functor (and thus has exponential objects). We denote that category as $\mathbf{FFG}_{\mathcal{S}}$. The morphisms in this category we refer to as $\mathsf{Arr}(\mathbf{FFG}_{\mathcal{S}})$ and its objects as $\mathsf{Obj}(\mathbf{FFG}_{\mathcal{S}})$.

Theorem 3.2.7. *Given $\mathcal{M}$, $\mathcal{N}$ ffg semimodules with bases M and N, the dual base of*

$$B = \{b_{\mu\nu} : \mu \in M, \nu \in N\} \subseteq Hom(\mathcal{M}, \mathcal{N})$$

is given by

$$B^* = \{b^*_{\mu\nu} : \mu \in M, \nu \in N\}$$

Where

$$b^*_{\mu\nu}(f) = \nu^*(f(\mu))$$

Proof. Notice that

$$b^*_{\mu\nu}(b_{\mu'\nu'}) = \nu^*(b_{\mu'\nu'}(\mu))$$

If $\mu = \mu'$ and $\nu = \nu'$ then this is 1. If $\mu \neq \mu'$ then $b_{\mu'\nu'}(\mu) = 0$ and if $\nu \neq \nu'$ then $\nu^*(b_{\mu'\nu'}(\mu)) = 0$ because $b_{\mu'\nu'}(\mu)$ is a multiple of ν'. □

Theorem 3.2.8. *Let $\mathcal{M}, \mathcal{N}$ and $\mathcal{P}$ be ffg semimodules with bases M, N and P respectively, and consider the space*

$$Hom(Hom(\mathcal{M}, \mathcal{N}), \mathcal{P})$$

Then the set

$$C = \{f \mapsto \nu^*(f(\mu)) \cdot \pi \ : \ \mu \in M, \nu \in N, \pi \in P\}$$

is a base of $Hom(Hom(\mathcal{M}, \mathcal{N}), \mathcal{P})$. In particular, the space is generated by elements of the form

$$f \mapsto g(f(m)) \quad m \in \mathcal{M}, g \in Hom(\mathcal{N}, \mathcal{P})$$

Proof. Let M, N and P be the bases of $\mathcal{M}$, $\mathcal{N}$ and $\mathcal{P}$ respectively. By theorem 3.2.6, the induced base for $\mathrm{Hom}(\mathcal{M}, \mathcal{N})$ is

$$B = \{b_{\mu\nu} : \mu \in M, \nu \in N\}$$

With dual base

$$B^* = \{b^*_{\mu\nu} : \mu \in M, \nu \in N\}$$

Applying theorem 3.2.6 again, the base of $\mathrm{Hom}(\mathrm{Hom}(\mathcal{M}, \mathcal{N}), \mathcal{P})$ is

$$C = \{c_{b_{\mu\nu}\pi} : \mu \in M, \nu \in N, \pi \in P\}$$

The elements of C are of the form

$$c_{b_{\mu\nu}\pi} \ = \ f \mapsto b^*_{\mu\nu}(f) \cdot \pi \ = \ f \mapsto \nu^*(f(\mu)) \cdot \pi$$

Which is of the required form, as the mapping

$$x \mapsto \nu^*(x) \cdot \pi$$

is a morphism. □

3.3 Tensor Product

It will be fundamental to the proposed denotational semantics to have an adjunction to the Hom functor. We define the tensor product of semimodules based on this adjunction.

Definition 3.3.1 (Tensor product)**.** *Given two semimodules $\mathcal{M}, \mathcal{N}$, a semimodule $\mathcal{T}$ together with a morphism $f : Hom(\mathcal{M}, Hom(\mathcal{N}, \mathcal{T}))$ is a tensor product if and only if for every semimodule $\mathcal{V}$ there exists an isomorphism*

$$\Psi : Hom(Hom(\mathcal{M}, Hom(\mathcal{N}, \mathcal{V})), Hom(\mathcal{T}, \mathcal{V}))$$

such that

$$\Psi(g)(f(m)(n)) = g(m)(n) \quad \text{for all} \quad g \in Hom(\mathcal{M}, Hom(\mathcal{N}, \mathcal{V})).$$

If they exist, tensor products are unique (up to isomorphism), because they are defined by a universal property. We can thus write $\mathcal{M} \otimes \mathcal{N}$ for the unique tensor product of $\mathcal{M}$ and $\mathcal{N}$, with $f(m)(n) = m \otimes n$ the associated morphism.

Theorem 3.3.2. *Let $\mathcal{M}, \mathcal{N}$ be ffg semimodules, then they have a ffg tensor product.*

Proof. Let M be a base of $\mathcal{M}$ and N a base of $\mathcal{N}$. Then the tensor product is the space

$$\mathcal{M} \otimes \mathcal{N} = \mathcal{S}^{M \times N}$$

Of formal linear combinations of pairs of elements of M and N.

The function $\otimes$ is given by the unique morphism extension of

$$\mu \otimes \nu = (\mu, \nu) \quad \mu \in M, \nu \in N$$

to $\mathcal{M}$ and $\mathcal{N}$.

Consider the morphism given by

$$\Psi : \mathrm{Hom}(\mathrm{Hom}(\mathcal{M}, \mathrm{Hom}(\mathcal{N}, \mathcal{V})), \mathrm{Hom}(\mathcal{T}, \mathcal{V}))$$

$$\Psi(f)\left(\sum_{(\mu,\nu)\in M\times N} s_{(\mu,\nu)} \cdot (\mu, \nu)\right) = \sum_{(\mu,\nu)\in M\times N} s_{(\mu,\nu)} \cdot f(\mu)(\nu)$$

with inverse defined as the unique extension of

$$\Psi^{-1}(f)(\mu)(\nu) = f((\mu, \nu)) \quad \mu \in M, \nu \in N$$

Notice that they are indeed inverses of each other, as

$$\begin{aligned}\Psi(\Psi^{-1}(f))\left(\sum_{(\mu,\nu)\in M\times N} s_{(\mu,\nu)}\cdot(\mu,\nu)\right) &= \sum_{(\mu,\nu)\in M\times N} s_{(\mu,\nu)}\cdot\Psi^{-1}(f)(\mu)(\nu)\\ &= \sum_{(\mu,\nu)\in M\times N} s_{(\mu,\nu)}\cdot f((\mu,\nu))\\ &= f\left(\sum_{(\mu,\nu)\in M\times N} s_{(\mu,\nu)}\cdot(\mu,\nu)\right)\end{aligned}$$

And conversely

$$\Psi^{-1}(\Psi(f))(\mu)(\nu) = \Psi(f)((\mu,\nu)) = f(\mu)(\nu)$$

□

4. TYPES, TERMS AND REDUCTIONS

4.1 Language

For a fixed semiring $\mathcal{S}$, the language $\mathcal{L}^{\mathcal{S}}$ proof system (or simply $\mathcal{L}$) was defined in [DCD24]. The types of the language are those of IMALL (Intuitionistic, Multiplicative-Additive Linear Logic), and are defined recursively as:

$$A = \mathbb{1} \mid A \multimap A \mid A \otimes A \mid \top \mid \mathbb{0} \mid A \,\&\, A \mid A \oplus A$$

Let us call the set of these types $\mathcal{P}$.

The proof-terms of the $\mathcal{L}$-calculus are defined to be

$$\begin{aligned} t = {} & x \mid t \boldsymbol{+} u \mid a \bullet t \\ & \mid a.\star \mid \delta_{\mathbb{1}}(t,u) \mid \lambda x.t \mid t\ u \mid t \otimes u \mid \delta_{\otimes}(t, xy.u) \\ & \mid \langle\rangle \mid \delta_{\mathbb{0}}(t) \mid \langle t,u\rangle \mid \delta^1_{\&}(t,x.u) \mid \delta^2_{\&}(t,x.u) \mid \mathsf{inl}(t) \mid \mathsf{inr}(t) \mid \delta_{\oplus}(t,x.u,y.v) \end{aligned}$$

where $a \in \mathcal{S}$ is a scalar.

The typing rules for the $\mathcal{L}$-calculus are given in Figure 4.1. The reduction rules are given in Figure 4.2. In [DCD24] it has been proven that the language is confluent and strongly terminating, and that it enjoys the subject reduction property.

We also define a subset $\mathcal{L}' \subseteq \mathcal{L}$, which is defined as the language that can be typed with the reduced set of types given by

$$A = \mathbb{1} \mid A \multimap A \mid A \otimes A \mid A \,\&\, A \mid A \oplus A$$

Let us call this reduced set of types $\mathcal{P}'$.

And its proof-terms are defined as

$$\begin{aligned} t = {} & x \mid t \boldsymbol{+} u \mid a \bullet t \\ & \mid a.\star \mid \delta_{\mathbb{1}}(t,u) \mid \lambda x.t \mid t\ u \mid t \otimes u \mid \delta_{\otimes}(t, xy.u) \\ & \mid \langle t,u\rangle \mid \delta^1_{\&}(t,x.u) \mid \delta^2_{\&}(t,x.u) \mid \mathsf{inl}(t) \mid \mathsf{inr}(t) \mid \delta_{\oplus}(t,x.u,y.v) \end{aligned}$$

4.2 Elimination Contexts

To define notions of computational equivalence between terms, we need to understand how terms reduce when replaced into other terms. To that end, two classes are considered:

$$\frac{}{x : A \vdash x : A}\ \text{ax} \qquad \frac{\Gamma \vdash t : A \quad \Gamma \vdash u : A}{\Gamma \vdash t \mathbin{+} u : A}\ \text{sum} \qquad \frac{\Gamma \vdash t : A}{\Gamma \vdash a \bullet t : A}\ \text{prod}(a)$$

$$\frac{}{\vdash a.\star : \mathbb{1}}\ \mathbb{1}\text{-i}(a) \qquad \frac{\Gamma \vdash t : \mathbb{1} \quad \Delta \vdash u : A}{\Gamma, \Delta \vdash \delta_{\mathbb{1}}(t, u) : A}\ \mathbb{1}\text{-e}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B}\ \multimap\text{-i} \qquad \frac{\Gamma \vdash t : A \multimap B \quad \Delta \vdash u : A}{\Gamma, \Delta \vdash t\ u : B}\ \multimap\text{-e}$$

$$\frac{\Gamma \vdash t : A \quad \Delta \vdash u : B}{\Gamma, \Delta \vdash t \otimes u : A \otimes B}\ \otimes\text{-i} \qquad \frac{\Gamma \vdash t : A \otimes B \quad \Delta, x : A, y : B \vdash u : C}{\Gamma, \Delta \vdash \delta_{\otimes}(t, xy.u) : C}\ \otimes\text{-e}$$

$$\frac{}{\Gamma \vdash \langle\rangle : \mathbb{o}}\ \mathbb{o}\text{-i} \qquad \frac{\Gamma \vdash t : \mathbb{o}}{\Gamma, \Delta \vdash \delta_{\mathbb{o}}(t) : C}\ \mathbb{o}\text{-e}$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash u : B}{\Gamma \vdash \langle t, u\rangle : A \mathbin{\&} B}\ \&\text{-i}$$

$$\frac{\Gamma \vdash t : A \mathbin{\&} B \quad \Delta, x : A \vdash u : C}{\Gamma, \Delta \vdash \delta^1_{\&}(t, x.u) : C}\ \&\text{-e1} \qquad \frac{\Gamma \vdash t : A \mathbin{\&} B \quad \Delta, x : B \vdash u : C}{\Gamma, \Delta \vdash \delta^2_{\&}(t, x.u) : C}\ \&\text{-e2}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathsf{inl}(t) : A \oplus B}\ \oplus\text{-i1} \qquad \frac{\Gamma \vdash t : B}{\Gamma \vdash \mathsf{inr}(t) : A \oplus B}\ \oplus\text{-i2}$$

$$\frac{\Gamma \vdash t : A \oplus B \quad \Delta, x : A \vdash u : C \quad \Delta, y : B \vdash v : C}{\Gamma, \Delta \vdash \delta_{\oplus}(t, x.u, y.v) : C}\ \oplus\text{-e}$$

Fig. 4.1: The typing rules of the $\mathcal{L}^{\mathcal{S}}$-calculus.

Definition 4.2.1. *Let $\mathcal{H}(A, B)$ be the set of terms η such that $\_ : A \vdash \eta : B$, where $\_$ is a distinguished variable (the "*hole*"). This is the class of terms we replace* into. *For convenience, $\mathcal{H}(A)$ is defined to be $\mathcal{H}(A, \mathbb{1})$.*

Definition 4.2.2. *Let $\mathcal{C}(A)$ be the set of closed terms t such that $\vdash t : A$. This is the class of terms we replace into other terms.*

If $t \in \mathcal{C}(A)$ and $\eta \in \mathcal{H}(A, B)$, then $(t/\_)\eta \in \mathcal{C}(B)$. Similarly, if $\eta \in \mathcal{H}(A, B)$ and $\psi \in \mathcal{H}(B, C)$, then $(\eta/\_)\psi \in \mathcal{H}(A, C)$. Based on this, we define:

$$@\colon \mathcal{C}(A) \times \mathcal{H}(A, B) \to \mathcal{C}(B), \quad t @ \eta = (t/\_)\eta$$

$$@\colon \mathcal{H}(A, B) \times \mathcal{H}(B, C) \to \mathcal{H}(A, C), \quad \eta @ \psi = (\eta/\_)\psi$$

It is clear that $@$ is associative, because it is term substitution.

Definition 4.2.3 (Elimination context [DCD24])**.** *We define $\mathcal{E}(A, B) \subseteq \mathcal{H}(A, B)$ as the subset of terms in the following language:*

$$\begin{aligned}\varepsilon = \_ \mid\ & \delta_{\mathbb{1}}(\varepsilon, u) \mid \varepsilon\ u \mid \delta_{\otimes}(\varepsilon, xy.v) \\ & \mid \delta^1_{\&}(\varepsilon, x.r) \mid \delta^2_{\&}(\varepsilon, x.r) \mid \delta_{\oplus}(\varepsilon, x.r, y.s)\end{aligned}$$

where u is a closed proof, $FV(v) = \{x, y\}$, $FV(r) \subseteq \{x\}$, and $FV(s) \subseteq \{y\}$. As with $\mathcal{H}$, $\mathcal{E}(A)$ is defined to be $\mathcal{E}(A, \mathbb{1})$.

$$
\begin{aligned}
\delta_{\mathbf{1}}(a.\star, t) &\longrightarrow a \bullet t \\
(\lambda x.t)\ u &\longrightarrow (u/x)t \\
\delta_{\otimes}(u \otimes v, xy.w) &\longrightarrow (u/x, v/y)w \\
\delta^1_{\&}(\langle t, u\rangle, x.v) &\longrightarrow (t/x)v \\
\delta^2_{\&}(\langle t, u\rangle, x.v) &\longrightarrow (u/x)v \\
\delta_{\oplus}(\mathsf{inl}(t), x.v, y.w) &\longrightarrow (t/x)v \\
\delta_{\oplus}(\mathsf{inr}(u), x.v, y.w) &\longrightarrow (u/y)w
\end{aligned}
$$

$$
\begin{aligned}
a.\star \mathbin{+} b.\star &\longrightarrow (a + b).\star \\
(\lambda x.t) \mathbin{+} (\lambda x.u) &\longrightarrow \lambda x.(t \mathbin{+} u) \\
\delta_{\otimes}(t \mathbin{+} u, xy.v) &\longrightarrow \delta_{\otimes}(t, xy.v) \mathbin{+} \delta_{\otimes}(u, xy.v) \\
\langle\rangle \mathbin{+} \langle\rangle &\longrightarrow \langle\rangle \\
\langle t, u\rangle \mathbin{+} \langle v, w\rangle &\longrightarrow \langle t \mathbin{+} v, u \mathbin{+} w\rangle \\
\delta_{\oplus}(t \mathbin{+} u, x.v, y.w) &\longrightarrow \delta_{\oplus}(t, x.v, y.w) \mathbin{+} \delta_{\oplus}(u, x.v, y.w)
\end{aligned}
$$

$$
\begin{aligned}
a \bullet b.\star &\longrightarrow (a \times b).\star \\
a \bullet \lambda x.t &\longrightarrow \lambda x.a \bullet t \\
\delta_{\otimes}(a \bullet t, xy.v) &\longrightarrow a \bullet \delta_{\otimes}(t, xy.v) \\
a \bullet \langle\rangle &\longrightarrow \langle\rangle \\
a \bullet \langle t, u\rangle &\longrightarrow \langle a \bullet t, a \bullet u\rangle \\
\delta_{\oplus}(a \bullet t, x.v, y.w) &\longrightarrow a \bullet \delta_{\oplus}(t, x.v, y.w)
\end{aligned}
$$

Fig. 4.2: The reduction rules of the $\mathcal{L}^{\mathcal{S}}$-calculus.

5. COMPUTATIONAL EQUIVALENCE

5.1 Weak and Strong Computational Equivalence

Recall that $\longleftrightarrow^*$ denotes the symmetric-transitive closure of the reduction rules, i.e. $t_1 \longleftrightarrow^* t_2$ if and only if both terms reduce to a common reduct.

Definition 5.1.1 (Weak Computational equivalence [DCD24])**.** *Two terms $u, v \in \mathcal{C}(A)$ are said to be* weakly *computationally equivalent ($u \equiv_w v$) if $u \mathbin{@} \varepsilon \longleftrightarrow^* v \mathbin{@} \varepsilon$ for all elimination contexts $\varepsilon \in \mathcal{E}(A)$.*

Definition 5.1.2 (Strong Computational equivalence)**.** *Two terms $u, v \in \mathcal{C}(A)$ are said to be* strongly *computationally equivalent ($u \equiv_s v$) if $u \mathbin{@} \eta \longleftrightarrow^* v \mathbin{@} \eta$ for all $\eta \in \mathcal{H}(A)$.*

Note that definitions 5.1.1 and 5.1.2 differ only in the set of terms that u and v can be replaced into. While 5.1.2 allows any term of the appropriate type, 5.1.1 only allows elimination contexts.

5.2 Proof of Equivalence

The objective of this section is to prove that the notions of weak and strong equivalence coincide (Theorem 5.2.5).

The implication $t_1 \equiv_w t_2 \;\Rightarrow\; t_1 \equiv_s t_2$ follows directly from the fact that 5.1.2 considers more terms than 5.1.1. To prove the converse, we will use two lemmas from [DCD24].

Lemma 5.2.1. *(Lemma 4.8 from [DCD24]) If $\eta \in \mathcal{H}(A, B)$ is an irreducible proof, then there exists a type C, an elimination context $\varepsilon \in \mathcal{E}(C, B)$, and a proof $w \in \mathcal{H}(A, C)$ such that:*

- $\eta = w \mathbin{@} \varepsilon$
- *w is either $\_$, an introduction, a sum, or a product.*

Lemma 5.2.2. *(Lemma 4.9 from [DCD24]) If $\varepsilon \in \mathcal{E}(A, B)$ and $\varepsilon \neq \_$, then there exists a type C, and elimination contexts $\varepsilon_1 \in \mathcal{E}(A, C)$, $\varepsilon_2 \in \mathcal{E}(C, B)$ such that $\varepsilon = \varepsilon_1 \mathbin{@} \varepsilon_2$ and ε_1 is an elimination of A.*

Definition 5.2.3. *Let A, B be types, $\eta \in \mathcal{H}(A, B)$ and $\varepsilon \in \mathcal{E}(B)$. Then we define*

$$P(\eta, \varepsilon) \;:=\; \forall t_1, t_2 \in \mathcal{C}(A)\;\; t_1 \equiv_w t_2 \Rightarrow t_1 \mathbin{@} \eta \mathbin{@} \varepsilon \longleftrightarrow^* t_2 \mathbin{@} \eta \mathbin{@} \varepsilon$$

Lemma 5.2.4 (Preservation of weak equivalence)**.** *Let A, B be types and $t_1, t_2 \in \mathcal{C}(A)$, $\eta \in \mathcal{H}(A, B)$, then*

$$t_1 \mathbin{@} \eta \equiv_w t_2 \mathbin{@} \eta$$

whenever

$$t_1 \equiv_w t_2$$

That is, we want to ensure that equivalent terms, when applied to the same context, yield equivalent terms.

Proof. Note that the condition

$$t_1 \mathbin{@} \eta \equiv_w t_2 \mathbin{@} \eta$$

in lemma 5.2.4 is equivalent to

$$\forall \varepsilon \in \mathcal{E}(B), \quad t_1 \mathbin{@} \eta \mathbin{@} \varepsilon \longleftrightarrow^* t_2 \mathbin{@} \eta \mathbin{@} \varepsilon.$$

Thus we want to prove that $P(\eta, \varepsilon)$ holds for each $\eta \in \mathcal{H}(A, B)$ and $\varepsilon \in \mathcal{E}(B)$.

We proceed by well-founded induction on the pair (η, ε), ordered lexicographically by the measure (r, e) where:

- r is the length of the longest *ultra-reduction* sequence starting from $\eta \mathbin{@} \varepsilon$, and
- e is the size of the term η.

Ultra-reduction, as defined in [DCD24], extends the reduction rules of the calculus with three additional rules: $t \mathbin{+} u \to t$, $t \mathbin{+} u \to u$, and $\sigma \bullet t \to t$. The value r is well defined because ultra-reduction is strongly terminating (cf. [DCD24]).

Note that if $\eta \longrightarrow^* \eta'$, then $P(\eta, \varepsilon) \iff P(\eta', \varepsilon)$, and moreover r does not increase (it may decrease by the reduction steps performed). Therefore, it suffices to prove $P(\eta, \varepsilon)$ for irreducible η.

Let us analyze all possible cases for η in the proposition $P(\eta, \varepsilon)$. Let t_1, t_2 be as in definition 5.2.3, then the possible cases for η according to lemma 5.2.2 are as follows:

Case $\eta = u \mathbin{+} v$

Since

$$t_i \mathbin{@} \eta = t_i \mathbin{@} (u \mathbin{+} v) = t_i \mathbin{@} u \mathbin{+} t_i \mathbin{@} v$$

Linearity implies that

$$t_i \mathbin{@} \eta \mathbin{@} \varepsilon = (t_i \mathbin{@} u \mathbin{+} t_i \mathbin{@} v) \mathbin{@} \varepsilon \longleftrightarrow^* t_i \mathbin{@} u \mathbin{@} \varepsilon \mathbin{+} t_i \mathbin{@} v \mathbin{@} \varepsilon$$

The inductive hypothesis applies to both (u, ε) and (v, ε): by the ultra-reduction rules $(u \mathbin{+} v) \mathbin{@} \varepsilon \to u \mathbin{@} \varepsilon$ and $(u \mathbin{+} v) \mathbin{@} \varepsilon \to v \mathbin{@} \varepsilon$, the first component r strictly decreases. Thus

$$t_1 \mathbin{@} u \mathbin{@} \varepsilon \longleftrightarrow^* t_2 \mathbin{@} u \mathbin{@} \varepsilon$$

and

$$t_1 \mathbin{@} v \mathbin{@} \varepsilon \longleftrightarrow^* t_2 \mathbin{@} v \mathbin{@} \varepsilon$$

By linearity, if we add those two equations we get back $P(\eta, \varepsilon)$.

Case $\eta = \sigma \bullet u$

Since

$$t_i \mathbin{@} \eta = t_i \mathbin{@} (\sigma \bullet u) = \sigma \bullet t_i \mathbin{@} u$$

Linearity implies that

$$t_i \mathbin{@} \eta \mathbin{@} \varepsilon = (\sigma \bullet t_i \mathbin{@} u) \mathbin{@} \varepsilon \longleftrightarrow^* \sigma \bullet t_i \mathbin{@} u \mathbin{@} \varepsilon$$

The inductive hypothesis applies to (u, ε): by the ultra-reduction rule $(\sigma \bullet u) \mathbin{@} \varepsilon \to u \mathbin{@} \varepsilon$, the first component r strictly decreases. Thus $t_1 \mathbin{@} u \mathbin{@} \varepsilon \longleftrightarrow^* t_2 \mathbin{@} u \mathbin{@} \varepsilon$. By linearity, that implies $P(\eta, \varepsilon)$.

Case $\eta = \lambda x.u$

Then B is a linear arrow type $B = P_1 \multimap P_2$. This means ε cannot be $\_$, since otherwise we would have $P_1 \multimap P_2 = \mathbb{1}$. By lemma 5.2.2, this means $\varepsilon = \varepsilon_2 \mathbin{@} \varepsilon_1$ where ε_2 is an elimination for $A \multimap B$. The only valid elimination for this type is application, so we have

$$\varepsilon_2 = \_ v$$

for some closed v. This means

$$\eta \mathbin{@} \varepsilon \longleftrightarrow^* \eta \mathbin{@} \varepsilon_2 \mathbin{@} \varepsilon_1 \longleftrightarrow^* (x/v)u \mathbin{@} \varepsilon_1$$

Note that the interaction of the introduction $\lambda x.u$ with the elimination ε_2 performs at least one reduction step (plus possibly additional steps to normalize the result to irreducible form). Therefore the first component r of the measure strictly decreases, and $P(\eta, \varepsilon)$ follows from $P((x/v)u, \varepsilon_1)$ by the inductive hypothesis.

Case $\eta = u \otimes v$

By a similar analysis, $\varepsilon = \varepsilon_2 \mathbin{@} \varepsilon_1$ where $\varepsilon_2 = \delta_\otimes(\_, xy.w)$. In particular,

$$\eta \mathbin{@} \varepsilon \longleftrightarrow^* \eta \mathbin{@} \delta_\otimes(\_, xy.w) \mathbin{@} \varepsilon_1 \longleftrightarrow^* (u/x)(v/y)w \mathbin{@} \varepsilon_1$$

As in the previous case, the interaction of the tensor introduction with the elimination ε_2 performs at least one reduction step, so the first component r strictly decreases and $P(\eta, \varepsilon)$ follows from $P((u/x)(v/y)w, \varepsilon_1)$.

Case $\eta = \langle u, v \rangle$

By a similar analysis, $\varepsilon = \varepsilon_2 \mathbin{@} \varepsilon_1$ where ε_2 is either $\delta^1_{\&}(\_, x.w)$ or $\delta^2_{\&}(\_, x.w)$. Both cases are analogous, so we will analyze the first one. We have that

$$\eta \mathbin{@} \varepsilon \longleftarrow^* \eta \mathbin{@} \varepsilon_2 \mathbin{@} \varepsilon_1 \longleftarrow^* \eta \mathbin{@} \delta^1_{\&}(\_, x.w) \mathbin{@} \varepsilon_1$$

$$\longleftarrow^* (u/x)w \mathbin{@} \varepsilon_1$$

Again, the interaction of the &-introduction with the elimination performs at least one reduction step, so r strictly decreases and $P(\eta, \varepsilon)$ follows from $P((u/x)w, \varepsilon_1)$.

Case $\eta = \mathsf{inl}(u)$

By a similar analysis, $\varepsilon = \varepsilon_2 \mathbin{@} \varepsilon_1$ where $\varepsilon_2 = \delta_{\oplus}(\_, x.v, y.w)$. We have

$$\eta \mathbin{@} \varepsilon \longleftarrow^* \eta \mathbin{@} \varepsilon_2 \mathbin{@} \varepsilon_1$$

$$\longleftarrow^* \eta \mathbin{@} \delta_{\oplus}(\_, x.v, y.w) \mathbin{@} \varepsilon_1$$

$$\longleftarrow^* (u/x)v \mathbin{@} \varepsilon_1$$

The interaction of inl with the $\oplus$-elimination performs at least one reduction step, so r strictly decreases and $P(\eta, \varepsilon)$ follows from $P((u/x)v, \varepsilon_1)$. Case $\eta = \mathsf{inr}(u)$ is analogous.

Case $\eta = \eta' \mathbin{@} \varepsilon'$ for elimination context ε'.

In this case note that $\eta \mathbin{@} \varepsilon = \eta' \mathbin{@} \varepsilon' \mathbin{@} \varepsilon$ so $P(\eta, \varepsilon) \iff P(\eta', \varepsilon' \mathbin{@} \varepsilon)$. Since η' is strictly smaller than η while r is unchanged, the inductive hypothesis applies (the second component e strictly decreases).

Case $\eta = \_$

This case follows directly from the definition of weak equivalence. □

Theorem 5.2.5 (Equivalence of weak and strong computational equivalence)**.** *Let T be a type and $t_1, t_2 \in \mathcal{C}(A)$ two terms, then*

$$t_1 \equiv_w t_2 \iff t_1 \equiv_s t_2$$

Proof. If two terms t_1 and t_2 are weakly equivalent, by lemma 5.2.4 they will remain weakly equivalent when applied to any extended elimination context. Extended elimination contexts are of type $\mathtt{1}$, and weak equivalence on closed terms of type $\mathtt{1}$ is the same as coreduction, which means that when t_1 and t_2 are applied to any extended elimination context, the resulting terms coreduce, meaning that t_1 and t_2 are *strongly* equivalent. □

From now on, let $\equiv$ be the equivalence relation $\equiv_s$ or $\equiv_w$, which we now know are equal.

Definition 5.2.6 (Extension of equivalence to non-closed terms)**.** *For a pair of non-closed terms t_1 and t_2 typed by*

$$x_1 : P_1, x_2 : P_2, \ldots, x_n : P_n \vdash t_{1 \leq i \leq 2} : P,$$

they are said to be computationally equivalent if and only if for any set of interpretations

$$c_1 \in \mathcal{C}(P_1), c_2 \in \mathcal{C}(P_2), \ldots, c_n \in \mathcal{C}(P_n)$$

The terms

$$t_{1 \leq i \leq 2}\{c_1/x_1, c_2/x_2, \ldots, c_n/x_n\}$$

are computationally equivalent.

6. DENOTATIONAL SEMANTICS

6.1 Definition of the Denotational Semantics

We will compare two semantics for the language $\mathcal{L}$: the operational semantics, denoted $(\!|.|\!)$; and the denotational semantics, denoted $[\![.]\!]$.

The denotational semantics is defined in [DCM23] and reproduced here for convenience.

Definition 6.1.1 (Denotational semantics for types)**.** *:*

- $[\![\mathbb{1}]\!] = \mathcal{S}$
- $[\![A \multimap B]\!] = \hom([\![A]\!], [\![B]\!])$
- $[\![A \oplus B]\!] = [\![A]\!] \times [\![B]\!]$
- $[\![A \& B]\!] = [\![A]\!] \times [\![B]\!]$
- $[\![A \otimes B]\!] = [\![A]\!] \otimes [\![B]\!]$

Note that by structural induction, $[\![A]\!]$ is always in $\mathbf{FFG}_{\mathcal{S}}$ and therefore the corresponding tensor products exist.

Every term $x_1 : A_1, x_2 : A_2, \cdots, x_n : A_n \vdash t : B$ is given a representation in the space

$$[\![t]\!] : \mathrm{Hom}\left(\bigotimes_{i=1}^{n} [\![A_i]\!], [\![B]\!]\right)$$

Lemma 6.1.2 (Maps)**.** *The following maps are in* $\mathsf{Arr}(\boldsymbol{FFG}_{\mathcal{S}})$*, for whenever*

$$A, B, C, D \in \mathsf{Obj}(\boldsymbol{FFG}_{\mathcal{S}}).$$

1. $A \xrightarrow{\hat{s}} A$ *defined by* $a \mapsto sa$.
2. $A \xrightarrow{\delta} A \times A$ *defined by* $a \mapsto (a, a)$.
3. $A \times A \xrightarrow{+} A$ *defined by* $(a_1, a_2) \mapsto a_1 +_A a_2$.
4. $\mathcal{S} \otimes A \xrightarrow{\bullet} A$ *defined by* $\sum_i s_i \otimes a_i \mapsto \sum_i s_i a_i$.
5. $A \xrightarrow{0} B$ *defined by* $a \mapsto 0_B$.
6. $(A \times B) \otimes C \xrightarrow{d} (A \otimes C) \times (B \otimes C)$*, defined by* $\sum_i (a_i, b_i) \otimes c_i \mapsto \sum_i (a_i \otimes c_i, b_i \otimes c_i)$.

7. $C \otimes (A \times B) \xrightarrow{d_r} (C \otimes A) \times (C \otimes B)$ *defined analogously.*
8. $A \xrightarrow{\eta^B} Hom(B, A \otimes B)$ *defined by* $a \mapsto (b \mapsto a \otimes b)$.
9. $Hom(A, B) \otimes A \xrightarrow{\varepsilon} B$ *defined by* $\sum_i f_i \otimes a_i \mapsto \sum_i f_i(a_i)$.
10. $A \xrightarrow{i_1} A \times B$ *defined by* $a \mapsto (a, 0)$
11. $B \xrightarrow{i_2} A \times B$ *defined by* $b \mapsto (0, b)$
12. $A \times B \xrightarrow{[u,v]} C$ *defined by* $(a, b) \mapsto u(a) + v(b)$ *whenever* $u : A \to C$ *and* $v : B \to C$

Definition 6.1.3 (Interpretation of proof-terms)**.** *We consider the following interpretation of proof-terms in* $\mathsf{Arr}(\boldsymbol{FFG}_{\mathcal{S}})$*. Since the deduction system is syntax directed, we give instead an interpretation for each deduction rule.*

- $[\![\dfrac{}{x : A \vdash x : A}\ \text{ax}]\!] = [\![A]\!] \xrightarrow{\mathsf{Id}} [\![A]\!]$
- $\left[\!\!\left[\dfrac{\Gamma \vdash t : A \quad \Gamma \vdash u : A}{\Gamma \vdash t + u : A}\ \text{sum}\right]\!\!\right] = [\![\Gamma]\!] \xrightarrow{\delta} [\![\Gamma]\!] \times [\![\Gamma]\!] \xrightarrow{t \times u} [\![A]\!] \times [\![A]\!] \xrightarrow{+} [\![A]\!]$
- $\left[\!\!\left[\dfrac{\Gamma \vdash t : A}{\Gamma \vdash s \bullet t : A}\ \text{prod}(s)\right]\!\!\right] = [\![\Gamma]\!] \xrightarrow{t} [\![A]\!] \xrightarrow{\hat{s}} [\![A]\!]$
- $[\![\dfrac{}{\vdash s.* : \mathbf{1}}\ \text{1-i}(s)]\!] = \mathcal{S} \xrightarrow{\hat{s}} \mathcal{S}$
- $\left[\!\!\left[\dfrac{\Gamma \vdash t : \mathbf{1} \quad \Delta \vdash u : A}{\Gamma, \Delta \vdash \delta_{\mathbf{1}}(t, u) : A}\ \text{1-e}\right]\!\!\right] = [\![\Gamma]\!] \otimes [\![\Delta]\!] \xrightarrow{t \otimes u} \mathcal{S} \otimes [\![A]\!] \xrightarrow{\bullet} [\![A]\!]$
- $[\![\dfrac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B}\ \multimap\text{-i}]\!]$
  $= [\![\Gamma]\!] \xrightarrow{\eta^{[\![A]\!]}} \mathsf{hom}([\![A]\!], [\![\Gamma]\!] \otimes [\![A]\!]) \xrightarrow{\mathsf{hom}([\![A]\!], t)} \mathsf{hom}([\![A]\!], [\![B]\!])$
- $\left[\!\!\left[\dfrac{\Gamma \vdash t : A \multimap B \quad \Delta \vdash u : A}{\Gamma, \Delta \vdash tu : B}\ \multimap\text{-e}\right]\!\!\right] =$
  $[\![\Gamma]\!] \otimes [\![\Delta]\!] \xrightarrow{t \otimes u} \mathsf{hom}([\![A]\!], [\![B]\!]) \otimes [\![A]\!] \xrightarrow{\varepsilon} [\![B]\!]$
- $\left[\!\!\left[\dfrac{\Gamma \vdash t : A \quad \Gamma \vdash u : B}{\Gamma \vdash \langle t, u \rangle : A \,\&\, B}\ \&\text{-i}\right]\!\!\right] = [\![\Gamma]\!] \xrightarrow{\delta} [\![\Gamma]\!] \times [\![\Gamma]\!] \xrightarrow{t \times u} [\![A]\!] \times [\![B]\!]$
- $\left[\!\!\left[\dfrac{\Gamma \vdash t : A \,\&\, B}{\Gamma \vdash \pi_1(t) : A}\ \&\text{-e1}\right]\!\!\right] = [\![\Gamma]\!] \xrightarrow{t} [\![A]\!] \times [\![B]\!] \xrightarrow{\pi_1} [\![A]\!]$
- $\left[\!\!\left[\dfrac{\Gamma \vdash t : A \,\&\, B}{\Gamma \vdash \pi_2(t) : B}\ \&\text{-e2}\right]\!\!\right] = [\![\Gamma]\!] \xrightarrow{t} [\![A]\!] \times [\![B]\!] \xrightarrow{\pi_2} [\![B]\!]$
- $\left[\!\!\left[\dfrac{\Gamma \vdash t : A, \Delta \vdash u : B}{\Gamma, \Delta \vdash t \otimes u : A \otimes B}\ \otimes\text{-i}\right]\!\!\right] = [\![\Gamma]\!] \otimes [\![\Delta]\!] \xrightarrow{t \otimes u} [\![A]\!] \otimes [\![B]\!]$
- $\left[\!\!\left[\dfrac{\Gamma \vdash t : A \otimes B \quad \Delta, x : A, y : B \vdash u : C}{\Gamma, \Delta \vdash \delta_{\otimes}(t, x.y.u) : C}\ \otimes\text{-e}\right]\!\!\right]$
  $= [\![\Gamma]\!] \otimes [\![\Delta]\!] \xrightarrow{t \otimes id} [\![A]\!] \otimes [\![B]\!] \otimes [\![\Delta]\!] \xrightarrow{u} [\![C]\!]$

- $\left[\!\left[\dfrac{\Gamma \vdash t : A}{\Gamma \vdash \mathsf{inl}\ t : A \oplus B}\ \oplus\text{-i1}\right]\!\right] = [\![\Gamma]\!] \xrightarrow{t} [\![A]\!] \xrightarrow{i_1} [\![A]\!] \times [\![B]\!]$
- $\left[\!\left[\dfrac{\Gamma \vdash t : B}{\Gamma \vdash \mathsf{inr} t : A \oplus B}\ \oplus\text{-i2}\right]\!\right] = [\![\Gamma]\!] \xrightarrow{t} [\![A]\!] \xrightarrow{i_2} [\![A]\!] \times [\![B]\!]$
- $\left[\!\left[\dfrac{\Gamma \vdash t : A \oplus B \quad x : A, \Delta \vdash u : C \quad y : B, \Delta \vdash v : C}{\Gamma, \Delta \vdash \delta_{\oplus}(t, x.u, y.v) : C}\ \oplus\text{-e}\right]\!\right]$
  $= [\![\Gamma]\!] \otimes [\![\Delta]\!] \xrightarrow{t \otimes \mathsf{Id}} ([\![A]\!] \times [\![B]\!]) \otimes [\![\Delta]\!] \xrightarrow{d} ([\![A]\!] \otimes [\![\Delta]\!]) \times ([\![B]\!] \otimes [\![\Delta]\!]) \xrightarrow{[u,v]} [\![C]\!]$

7. OPERATIONAL CATEGORY

7.1 Definition of the Operational Category

We define a category where the arrows are the terms in the language quotiented by computational equivalence

Definition 7.1.1. *Operational Category. Let the operational category be a category with an object* $(\!|P|\!)$ *for every type* $P \in \mathcal{P}$*, and an arrow* $f : Hom((\!|A|\!), (\!|B|\!))$ *for every term in* $\mathcal{H}(A, B)$ *modulo computational equivalence. The equivalence class of a term* $t \in \mathcal{H}(A, B)$ *is denoted as* $(\!|t|\!) : Hom((\!|A|\!), (\!|B|\!))$*.*

The composition is given by $(\!|w|\!) \circ (\!|t|\!) = (\!|t \mathbin{@} w|\!)$*.*

To prove the well definition of this category, we need to prove that the composition is well defined and that it is associative.

Lemma 7.1.2. *Composition is well defined on the right. Consider* $t, t' \in \mathcal{H}(A, B)$ *such that* $t \equiv t'$ *and* $w \in \mathcal{H}(B, C)$ *then* $t \mathbin{@} w = t' \mathbin{@} w$*.*

Proof. Consider any $a \in \mathcal{C}(A)$ and $\phi \in \mathcal{H}(C)$. By definition of computational equivalence, we need to prove

$$c \mathbin{@} t \mathbin{@} w \mathbin{@} \phi \longleftrightarrow^* c \mathbin{@} t' \mathbin{@} w \mathbin{@} \phi$$

But this follows from $t \equiv t'$, since both are replaced with the same term c and in the same context $w \mathbin{@} \phi$. □

Lemma 7.1.3. *Composition is well defined on the left. Consider* $w, w' \in \mathcal{H}(B, C)$ *such that* $w \equiv w'$ *and* $t \in \mathcal{H}(A, B)$ *then* $t \mathbin{@} w = t \mathbin{@} w'$*.*

Proof. Consider any $a \in \mathcal{C}(A)$ and $\phi \in \mathcal{H}(C)$. By definition of computational equivalence, we need to prove

$$c \mathbin{@} t \mathbin{@} w \mathbin{@} \phi \longleftrightarrow^* c \mathbin{@} t \mathbin{@} w' \mathbin{@} \phi$$

But this follows from $w \equiv w'$, since both are replaced with the same term $c \mathbin{@} t$ and in the same context ϕ. □

Lemma 7.1.4. *Composition is associative. Consider* $t \in \mathcal{H}(A, B)$*,* $u \in \mathcal{H}(B, C)$ *and* $v \in \mathcal{H}(C, D)$*, then* $t \mathbin{@} (u \mathbin{@} v) = (t \mathbin{@} u) \mathbin{@} v$*.*

Proof. The proof follows directly from the associativity of the substitution operation. □

7.2 Linearity of the Operational Category

Consider two types A and B, let us define a function parametrized by a $c \in \mathcal{C}(A)$ and $\phi \in \mathcal{H}(B)$

$$F_c^\phi : \mathcal{H}(A,B) \to \mathcal{S} \qquad f \mapsto s \text{ whenever } c \mathbin{@} f \mathbin{@} \phi \longleftrightarrow^* s.*$$

Where the $\longleftrightarrow^*$ relationship is the transitive closure of the reduction rules. We have that

Lemma 7.2.1. *Two terms t_1 and t_2 are not computationally equivalent if and only if there exists some F_c^ϕ such that $F_c^\phi(t_1) \neq F_c^\phi(t_2)$.*

Proof. Direct from the definition. □

Let us prove the linearity of the functions F_c^ϕ

Lemma 7.2.2. *Let $t_1, t_2 \in \mathcal{H}(A,B)$, $s \in \mathcal{S}$ and let $c \in \mathcal{C}(A), \phi \in \mathcal{H}(B)$, then*

$$F_c^\phi(s \bullet t_1 + t_2) = sF_c^\phi(t_1) + F_c^\phi(t_2)$$

Proof. For this, we can use the denotational semantics. Notice that for any closed term t of type $\mathtt{1}$, by strong normalization and subject reduction there exists a unique $s' \in \mathcal{S}$ such that $t \longleftrightarrow^* s'.*$. Since $[\![.]\!]$ is sound with respect to $\longleftrightarrow^*$ (i.e. it satisfies subject reduction, proven in [DCM23]), we have $[\![t]\!] = [\![s'.*]\!] = (x \mapsto s'x)$. Conversely, if $[\![t]\!] = (x \mapsto sx)$, then $t \longleftrightarrow^* s'.*$ for some s', and soundness gives $s = s'$. Therefore

$$c \mathbin{@} f \mathbin{@} \phi \longleftrightarrow^* s.* \quad \Leftrightarrow \quad [\![c \mathbin{@} f \mathbin{@} \phi]\!] = (x \mapsto sx)$$

and the equation we want to prove reduces to

$$[\![c \mathbin{@} (s \bullet t_1 + t_2) \mathbin{@} \phi]\!] = s\,[\![c \mathbin{@} t_1 \mathbin{@} \phi]\!] + [\![c \mathbin{@} t_2 \mathbin{@} \phi]\!]$$

Which follows directly from linearity of composition, and the fact that

$$[\![s \bullet t_1 + t_2]\!] = s\,[\![t_1]\!] + [\![t_2]\!]\,.$$

□

This linearity condition endows the space $\mathrm{Hom}(\langle\!|A|\!\rangle, \langle\!|B|\!\rangle)$, whenever it is nonempty, with a semimodular structure given by syntactical addition. Indeed, if we define

$$s\langle\!|t_1|\!\rangle + \langle\!|t_2|\!\rangle = \langle\!|s \bullet t_1 + t_2|\!\rangle$$

This addition and multiplication by scalar is not only well-defined, but also satisfies all the associativity and commutativity laws required for $\mathrm{Hom}(\langle\!|A|\!\rangle, \langle\!|B|\!\rangle)$ to form a semimodule, because any failure of those laws would translate, through a suitable F_c^ϕ function, to a failure of those laws on the semimodule $\mathcal{S}$.

Due to Lemma 7.2.1, the functions F_c^ϕ lift to the equivalence classes, yielding a function

$$F_c^\phi : \mathrm{Hom}(\langle\!|A|\!\rangle, \langle\!|B|\!\rangle) \to \mathcal{S}$$

and due to Lemma 7.2.2, these lifted functions are actually semimodule morphisms, whenever $\mathrm{Hom}(\llparenthesis A \rrparenthesis, \llparenthesis B \rrparenthesis)$ is nonempty. Therefore, the set

$$\{F_c^\phi : c \in \mathcal{C}(A), \phi \in \mathcal{H}(B)\}$$

Spans the dual space $\mathrm{Hom}(\llparenthesis A \rrparenthesis, \llparenthesis B \rrparenthesis)^* = \mathrm{Hom}(\mathrm{Hom}(\llparenthesis A \rrparenthesis, \llparenthesis B \rrparenthesis), \mathcal{S})$.

7.3 Object Semimodules

In the operational category, every pair of types $A, B \in \mathcal{P}$ gets an associated set $\mathrm{Hom}(\llparenthesis A \rrparenthesis, \llparenthesis B \rrparenthesis)$ which forms a semimodule. However, we would like to associate a semimodule to individual objects as well.

Definition 7.3.1. *Given a type $A \in \mathcal{P}$, we define $\llparenthesis A \rrparenthesis$ as the set of terms in $\mathcal{C}(A)$ quotiented by computational equivalence.*

Notice that there is an isomorphism between $\llparenthesis A \rrparenthesis$ and $\mathrm{Hom}(\llparenthesis \mathtt{1} \rrparenthesis, \llparenthesis A \rrparenthesis)$. Indeed, we have the equivalences

$$t \in \mathcal{C}(A) \quad \Leftrightarrow \quad \delta_{\mathtt{1}}(\_, t) \in \mathcal{H}(\mathtt{1}, A)$$
$$t \in \mathcal{H}(\mathtt{1}, A) \quad \Leftrightarrow \quad 1.* \mathbin{@} t \in \mathcal{C}(A)$$

Which is a bijection when quotiented by computational equivalence (or indeed by coreducibility).

8. RELATIONSHIP BETWEEN THE DENOTATIONAL AND OPERATIONAL SEMANTICS

8.1 Counterexample

It has been proven in [DCM23] that $[\![.]\!]$ satisfies subject reduction as well as substitution. These two properties when taken together imply soundness. That is, if $x, y \in \mathcal{C}(A)$ then $[\![x]\!] = [\![y]\!]$ implies $x \equiv y$.

Indeed, if we take any context $\varepsilon \in \mathcal{H}(A, 1)$, then by substitution $[\![x \mathbin{@} \varepsilon]\!] = [\![y \mathbin{@} \varepsilon]\!]$, but then if we take $x \mathbin{@} \varepsilon \longleftrightarrow^* a_x.*$ and $y \mathbin{@} \varepsilon \longleftrightarrow^* a_y.*$ (which exist by termination), then by subject reduction that implies that $[\![a_x.*]\!] = [\![a_y.*]\!]$ which implies $a_x = a_y$.

However, the converse is not true. Indeed,

Example 8.1.1. *There exists a $P \in \mathcal{P}$ and $x, y \in \mathcal{C}(P)$ such that $(\!|x|\!) = (\!|y|\!)$ but $[\![x]\!] \neq [\![y]\!]$.*

Proof. If we take x to be

$$\vdash 1 \bullet \lambda p.\pi_1(p) : (\mathbf{1} \& \mathbf{0}) \multimap \mathbf{1}$$

And y to be

$$\vdash 2 \bullet \lambda p.\pi_1(p) : (\mathbf{1} \& \mathbf{0}) \multimap \mathbf{1}$$

Then $[\![x]\!]$ and $[\![y]\!]$ are two distinct elements of the semimodule $\mathrm{Hom}(\mathcal{S}^2, \mathcal{S})$, but are nevertheless observationally equal, as it is impossible to compare the two functions by evaluating them because the domain $\mathbf{1} \& \mathbf{0}$ is uninhabited. □

We aim to prove that this property holds if we restrict to terms in $\mathcal{C}(P)$ for a type $P \in \mathcal{P}'$. Moreover, we wish to prove an isomorphism between the semimodules $(\!|A|\!)$ and $[\![A]\!]$ compatible with the interpretation of terms.

8.2 Denotational semantics are surjective

The function $(\!|.|\!) : \mathcal{C}(P) \to (\!|P|\!)$, which is defined whenever P is inhabited is surjective, as it is the canonical projection into a set of equivalence classes. If we want to establish an isomorphism, we need $[\![.]\!]$ to be surjective as well.

Theorem 8.2.1. *The function $[\![.]\!] : \mathcal{C}(P) \to [\![P]\!]$ is surjective whenever $P \in \mathcal{P}'$.*

Proof. We proceed by structural induction on P. Let us consider the cases:

- If $P = \mathbf{1}$ then for every $s \in \mathcal{S} = [\![\mathbf{1}]\!]$ we have $[\![s.*]\!] = s$, so we are done.

- For $P = A\&B$ we have that $[\![A\&B]\!] = [\![A]\!] \times [\![B]\!]$ and thus any point $x \in [\![P]\!]$ is of the form $\tau = (\alpha, \beta)$ with $\alpha \in [\![A]\!]$ and $\beta \in [\![B]\!]$. Let $a : A$ and $b : B$ such that $[\![a]\!] = \alpha$ and $[\![b]\!] = \beta$, which exist due to the inductive hypothesis. Then $[\![\langle a, b\rangle]\!] = \tau$.
- For $P = A \oplus B$ the construction is similar to the previous case, except that we take $\mathsf{inl}(a) + \mathsf{inr}(b)$ instead of $\langle a, b\rangle$.
- For $P = A \otimes B$ we know that $[\![A \otimes B]\!]$ is generated by the tensors of the form $\alpha \otimes \beta$, and each of those in turn is the image of $a \otimes b$ where $[\![a]\!] = \alpha$ and $[\![b]\!] = \beta$.
- For $P = A \multimap B$ we need to perform structural induction on the type A,
  - If $A = \mathtt{1}$ then the morphisms with $\mathcal{S}$ as domain are precisely those of the form $s \mapsto s \cdot \beta$ with $\beta \in [\![B]\!]$. Since β by the first inductive hypothesis has a preimage b, we can take $s \mapsto s \cdot \beta = [\![\lambda s.\delta_1(s, b)]\!]$.
  - If A is either $C\&D$ or $C \oplus D$ then $[\![A]\!] = [\![C]\!] \times [\![D]\!]$, and $\hom([\![A]\!], [\![B]\!]) = \hom([\![C]\!], [\![B]\!]) \times \hom([\![D]\!], [\![B]\!])$. Therefore any point in $\mathrm{Hom}([\![A]\!], [\![B]\!])$ can be written as a pair (γ, δ) and we have that $[\![c]\!] = \gamma$ and $[\![d]\!] = \delta$ by the second inductive hypothesis for some c and d. For the case $A = C\&D$ we can take $a = \lambda x.c\pi_1(x) \blacklozenge d\pi_2(x)$, and for $A = C \oplus D$ we can take $a = \lambda x.\delta_\oplus(x, y.cy, z.dz)$.
  - If $A = C \otimes D$ then let $\alpha : \hom([\![C]\!] \otimes [\![D]\!], [\![B]\!])$ and let α' be an element of the isomorphic $\hom([\![C]\!], \hom([\![D]\!], [\![B]\!]))$ such that $\alpha'(c)(d) = \alpha(c \otimes d)$. By the second inductive hypothesis there exists an $a \in \mathcal{C}(C \multimap D \multimap B)$ with $[\![a]\!] = \alpha'$. Then we can take $\alpha = [\![\lambda x.\delta_\otimes(x, cd.a(c)(d))]\!]$.
  - If $A = C \multimap D$, then every element of $\mathrm{Hom}(\mathrm{Hom}([\![C]\!], [\![D]\!]), [\![B]\!])$ is generated by elements of the form $f \mapsto \delta(f(\gamma))$ where $\gamma \in [\![C]\!]$ and $\delta \in \mathrm{Hom}([\![D]\!], [\![B]\!])$ by theorem 3.2.8. By the first inductive hypothesis there are $c \in \mathcal{C}(C)$ and $d \in \mathcal{C}(D \multimap B)$ such that $[\![c]\!] = \gamma$ and $[\![d]\!] = \delta$, and we can take $f \mapsto \delta(f(\gamma)) = [\![\lambda f.d(f(c))]\!]$. □

Remark 8.2.2. *Note that Theorem 8.2.1 implies that all types $P \in \mathcal{P}'$ are inhabited, as the corresponding semimodule $[\![P]\!]$ cannot, by definition, be empty.*

8.3 Main Results

Theorem 8.3.1. *Let $P \in \mathcal{P}'$, and let $t, u \in \mathcal{C}(P)$. If $(\!|t|\!) = (\!|u|\!)$, then $[\![t]\!] = [\![u]\!]$.*

Proof. Suppose $[\![t]\!] \neq [\![u]\!]$. Then there exists a function $\phi \in \hom([\![P]\!], \mathcal{S})$ that separates both values. That is, such that $\phi([\![t]\!]) \neq \phi([\![u]\!])$. Due to surjectivity of $[\![.]\!]$, we know there exists a proof $f \in \mathcal{C}(P \multimap \mathtt{1})$ that computes this function. But then we have that the terms t and u when evaluated under the (extended) context f must produce two different values, and therefore they must have different operational interpretations. □

Theorem 8.3.2. *There is a semimodule isomorphism between $(\!|P|\!)$ and $[\![P]\!]$, for any $P \in \mathcal{P}'$*

Proof. Let $P \in \mathcal{P}'$, and let $p \in (\!|P|\!)$. By definition, there exists some $t \in \mathcal{C}(P)$ such that $(\!|t|\!) = p$. We can then define $\phi(p) = [\![t]\!]$.

The well-definition of ϕ is given by 8.3.1. To prove that ϕ is a morphism, consider $t, u \in \mathcal{C}(P)$ and $s \in \mathcal{S}$. We need to prove that

$$\phi(s(\!|t|\!) + (\!|u|\!)) = s\phi((\!|t|\!)) + \phi((\!|u|\!))$$

$$\phi((\!|s \bullet t + u|\!)) = s\,[\![t]\!] + [\![u]\!]$$

$$\phi((\!|s \bullet t + u|\!)) = [\![s \bullet t + u]\!]$$

Which is true by definition. □

9. CONCLUSION & FUTURE WORK

9.1 Overview of the Results

The main results of this paper are

- Proving that the semantics for $\mathcal{L}^{\mathcal{S}}$ defined in [DCM23] is not sound with respect to computational equivalence.
- Defining a subset of well-behaved types and proving the semantics is sound on the associated sublanguage.
- Showing that the completeness property corresponds to an isomorphism between the denotational and operational categories.

9.2 Generalizations of the Semantics

One possible avenue of future work is to generalize the setting in which the semantics arises. For example, an abstract categorical construction could be used in place of the category of semimodules, or perhaps a laxer algebraic structure. In [DCM24] an example of such generalized semantics is proposed, not based on semimodular categories but on abstract properties. It is an open question whether soundness also fails in that case.

9.3 Generalizations of the Sound Types

In this thesis we define a specific subset of types for which we can prove that the semantics is sound, but we do not prove these are the only ones for which it is the case. For example, despite the type $\mathfrak{o}$ not being on $\mathcal{P}'$, the subset we defined as well-behaved, the semantics is trivially sound for that type.

It remains an open question to exactly characterize the types for which $[\![.]\!]$ is sound with respect to computational equivalence.

9.4 Achieving Full Abstraction

The natural question that arises from this work is if the $[\![.]\!]$ semantics can be tweaked to achieve full abstraction for all types. Some approaches we tried that attempted to bridge this gap involved slight changes to the category to be able to represent "empty semimodules" that could be surjectively mapped from uninhabited types.

However, no such small tweaks of that type can work since the problem of deciding whether a type in IMALL (in fact, in IMLL) is inhabited is NP-hard, as proven in [Lin95], so the current approach that encodes each type in a space of polynomial-size dimension would have to be fundamentally changed.

BIBLIOGRAPHY

[BS04] Richard Blute and Philip Scott. *Category Theory for Linear Logicians*, pages 3–64. London Mathematical Society Lecture Note Series. Cambridge University Press, 2004.

[DCD24] Alejandro Díaz-Caro and Gilles Dowek. A linear linear lambda-calculus. *Mathematical Structures in Computer Science*, 34(10):1103–1137, 2024.

[DCM23] Alejandro Díaz-Caro and Octavio Malherbe. Semimodules and the (syntactically-)linear lambda calculus. Draft at https://arxiv.org/abs/2205.02142v2, 2023.

[DCM24] Alejandro Díaz-Caro and Octavio Malherbe. The sup connective in imall: A categorical semantics. Draft at https://arxiv.org/abs/2205.02142v3, 2024.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[How80] William Alvin Howard. The formulae-as-types notion of construction. In Haskell Curry, Hindley B., Seldin J. Roger, and P. Jonathan, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*. Academic Press, 1980.

[Lam68] Joachim Lambek. Deductive systems and categories. *Mathematical systems theory*, 2:287–318, 1968.

[Lin95] Patrick Lincoln. Deciding provability of linear logic formulas. In *Proceedings of the Workshop on Advances in Linear Logic*, pages 109–122. Cambridge University Press, 1995.

[LS86] Joachim Lambek and Philip J. Scott. *Introduction to Higher Order Categorical Logic*, chapter 1.1-1.3. Cambridge University Press, 1986.