Mixing classical and intuitionistic constructiveness: a formulae-as-types interpretation for classical S4 modal logic and second-class continuations

Jean Caspar
Advisor: Guillaume Munch-Maccagnoni, INRIA, Nantes

August 20, 2025

General context

Proof assistants based on dependent type theory are becoming more and more popular today in the mathematics community as Mathlib and Lean draws more and more of them towards formalization, but even though the Curry-Howard correspondence has been extended for classical logic through the use of control operators [Fel+87] since [Gri89], mixing dependent types theory and constructive classical logic has not yet been achieved: indeed, it has been proved that naively mixing unrestricted dependent types and classical logic implies proof-irrelevance [Coq89]: all proofs are equal.

On the other hand, CPS translation allows to implement control operators, but also to compile efficiently. In 2019, Cong et al. [Con+19] devised a λ-calculus which is interesting in order to better understand the CPS translation: devised as an intermediate language for compilation, its types are annotated with a level, which is one or two; there is a $\mathcal{C}$ control operator which implements double negation elimination on level 2. In their calculus, every second-class term can be allocated on the stack; and first-class values do not refer to second-class variables in their closure, so the stack can be cleaned when a function accepts a first-class argument. This ensures an efficient compilation: continuations introduced by the compiler are all second-class, so they do not need more heap memory. The addition of second-class types do not increase the logical power, so the first level remains intuitionistic. Furthermore, their CPS translation presents intuitionistic features without being syntactically linear: continuations can be duplicated or forgotten to a certain extent.

Furthermore, on a seemingly completely different matter, S4 is a modal logic with an additional connector $\Box$. The Gödel-McKinsey-Tarski theorem says that the Gödel translation of intuitionistic logic into the S4, which essentially replaces $A \to B$ by $\Box(A \to B)$ recursively, preserves and reflects provability [Göd01; MT48].

Research problem

In order to build a dependent type theory with effects, it seems that the crucial property is the thunkability property, identified by [Füh99], which amounts to say that the term can be substituted safely — intuitively, this means that means that the term is pure. For example,

if t is thunkable then "$\lambda x.t$" and "let $y = t$ in $\lambda x.y$" behave the same; this is not true if, for example, t raises an exception or loops indefinitely.

The goal of this internship was to investigate the notion of second-class continuations for logic, to see if the the stackability property of second-class values highlighted by Cong et al. means that they are pure or thunkable in some sense, and to study if they do not increase the logical power of the calculus, thus allowing to mix an intuitionistic theory with some control. This will deepen our understanding of the Curry-Howard correspondence for classical logic, as well as the logical understanding of the difference of continuations used in compilation and in logic.

Your contribution

I have shown that the first-class arrow of Cong et al. can be decomposed as $\Box(A \to B)$, where $\to$ is the second-class arrow and $\Box$ is a modality which restricts the closure so that it only refers to first-class variables, and moreover, that terms of type $\Box A$ in a modal context, which corresponds to first-class terms, are thunkable, even when they use the $\mathcal{C}$ operator. Furthermore, we realized that the modality $\Box$ shares the laws of the S4 modality. Thus, based on these principles, I devised a $\bar{\lambda}\mu\tilde{\mu}$-calculus, the system $\mathbf{L}^{\Box}_{\mathsf{pol}}$, which implements polarized classical logic: there are positive (i.e. strict/call-by-value) and negative (i.e. lazy/call-by-name) versions of connectors like the disjunction: this allows a more fine-tuned operational semantics, and to adequately represent the distinction between the intuitionistic negation and the classical negation, the latter being negative; furthermore, there is a modality $\Box$ representing the S4 modality. I also have a λ-calculus version of this system, as well as a CPS translation, and the $\mathbf{L}^{\Box}_{\mathsf{pol}}$ system is shown to factorizes the CPS translation of Cong et al [Con+19].

My two main results are:

1. an observational model in which all terms of modal type in a modal context are thunkable;
2. a variant of the Gödel-McKinsey-Tarski theorem: intuitionistic logic can be embedded by translating $A \to B$ as $\Box(A \to B)$ and the translation preserves and reflects provability.

The system $\mathbf{L}^{\Box}_{\mathsf{pol}}$ shows that one can devise a classical calculus, with control effects, while keeping a large class of thunkable terms, into which the usual λ-calculus can be embedded.

Arguments supporting its validity

The unexpected connection with the S4 modal logic leads me to think that there is something more profound under the hood. I have written detailed proofs of all the claims I made in this report, mostly in appendices (see section A, section B and section C).

Summary and future work

I devised $\mathbf{L}^{\Box}_{\mathsf{pol}}$, a calculus for classical logic with control effects, and where all terms of "modal" type are thunkable; moreover, there is an interpretation of intuitionistic logic into modal types, so we may say that all terms of intuitionistic type are thunkable.

Following [Miq19], who build a dependent type theory with classical control, based on the fact that a specific fragment — the NEF fragment — was thunkable, the next goal would be to integrate dependent types in this system, where dependent types could only depend on modal types. The modal fragment is far bigger than the NEF fragment, because intuitionistic logic can be embedded into it, so the resulting type theory would be more powerful, maybe with the possibility to serve as a foundation for classical mathematics.

1. Introduction

In this report, I would like to thank both my advisor Guillaume Munch-Maccagnoni and the whole Gallinette team, especially Johann Rosain and Léo Soudant with whom I shared my office, as working with them for the few months I have been there was a great experience, as well as Étienne Miquey from I2M, who collaborated with us remotely and whom I had the chance of meeting during his short stay in Nantes.

Modern proof assistants are based constructive logic. Even though a computational interpretation of classical proofs exists through the use of control operators [Fel+87], dependent types have proven being difficult to mix with effects in general [Her05] and especially with classical logic [Coq89], thus, the excluded middle is only added as an axioms in proof assistants.

Having a dependent type theory with classical control may allow to better reason on CPS translation and typed compilation of dependently typed programs; moreover, it will correspond more accurately to the logic mathematician use in their daily work. A first step in that direction was solving the difficult problem of CPS translation for dependently typed programs, which have been achieved by [Bow18]. CPS translation are used in order to implement efficient compilation, but also allows to implement control operators by directly accessing the continuation.

Cong et al. [Con+19] presented a λ-calculus which has interesting properties: types are either level 1 or level 2, and the $\mathcal{C}$ operator implements double negation elimination with respect to level 2 negation, inspired by the original $\mathcal{C}$ operator of [Fel+87] which Griffin shown that it implements double negation elimination [Gri89]; moreover, level 1 values do not refer to level 2 variables, and level 2 values can be allocated on the stack. Moreover, thir CPS translation has intuitionistic features. I studied this stackability property and its relation with thunkable, as well as the notion of second-class continuation, in order to understand them from a logical point of view, and showed that their level 1 function type can be decomposed as $\Box(A \to B)$, where $\Box$ is a modality respecting the S4 laws, and moreover, I showed that terms of type $\Box$ defined in a modal context, or equivalently, first-class terms defined in a first class context are thunkable. Their calculus forbid functions to returns second-class elements; we lift this restriction, as explained in section D.

I will first present in section 2 the $\lambda_{\Box}\mathcal{C}$-calculus, a call-by-value, simply typed calculus with a S4-like modality $\Box$ and a $\mathcal{C}$ control operator implementing double negation elimination. I will then give in section 3 a translation of this calculus into a $\bar{\lambda}\mu\tilde{\mu}$-calculus, the system $\mathbf{L}^{\Box}_{\mathsf{pol}}$, which preserves preserves proofs, reductions, and types. The system $\mathbf{L}^{\Box}_{\mathsf{pol}}$ features a simple type system with a modality $\Box$. It is polarized, which means that connectors comes in two flavours: positive (i.e., strict/call-by-value) and negative (i.e., lazy/call-by-name). For example, the call-by-value arrow of the $\lambda_{\Box}\mathcal{C}$-calculus is defined as $\Downarrow(A \to B)$, with $\to$ the negative arrow, and $\Downarrow$ the positive polarity shift. Effects arise in the system $\mathbf{L}^{\Box}_{\mathsf{pol}}$ when a term captures a continuation and uses it non-linearly. This system is a $\bar{\lambda}\mu\tilde{\mu}$-calculus: these kinds of systems are more suitable than λ-calculus to work with classical logic, because they represent sequent calculus instead of natural deduction, which is ill-behaved for classical logic. After that, in section 4, I present a CPS translation on the system $\mathbf{L}^{\Box}_{\mathsf{pol}}$ which targets a fragment of the $\lambda_{\Box}\mathcal{C}$-calculus, and show that it factorizes the CPS translation of the $\lambda_{\Box}\mathcal{C}$-calculus. Then, I show that the system $\mathbf{L}^{\Box}_{\mathsf{pol}}$ is strongly normalizing in section 5. After that, I explain in section 6 what the thunkability property is in this setting: roughly, a thunkable term t is one that behaves like a value, i.e. "$\lambda x.t$" and "let $y = t$ in $\lambda x.y$" are equivalent. I provide a model of the system $\mathbf{L}^{\Box}_{\mathsf{pol}}$ in section 7,

based on observational equivalence, from which I can deduce the consistency of the system, and I also show my first main result: in this model, every term of modal type in a modal context is thunkable. In section 8 I show that proofs can be focused, which is nice technical property on sequents, used for the proof of the second main result. Finally in section 9 I prove my second main result, an analogue of the Gödel-McKinsey-Tarski theorem [Göd01; MT48]: the translation of intuitionistic logic into the system $\mathbf{L}^{\Box}_{\mathsf{pol}}$ preserves and reflects provability.

2. The $\lambda_{\Box}\mathcal{C}$-calculus

We introduce here a call-by-value λ-calculus for the classical S4 modal logic. This calculus is a call-by-value variant of the calculus of Kavvos [Kav17], to which I added a $\mathcal{C}$ control operator, implementing classical logic. The types are the types of the simply typed λ-calculus, with an additional unary connective, $\Box$ (read "box"). In S4, the axioms of $\Box$ make it a strong comonad. There are two typing judgments: $\Gamma \nmid \Theta \vdash t : A$ and $\Gamma \nmid \Theta \vdash t : \bot$. Γ is the usual context, but Θ consists of types B which are implicitly to be seen as $\Box B$. The two judgments are distinct because $\bot$ is not a real type; it can be used to form the type $\neg A$ because we have a primitive notion of negation, but we cannot build, for example, the type $A \otimes \bot$. This restriction is engraved in the syntax; for example, there are let expressions that return $\bot$ and let which return another type. This calculus is inspired by and refines the calculus of [Con+19]; see section D for more details.

We define a function $\varpi(A) \in \{+, \Box\}$ on types which returns its mode/polarity. If $\varpi(A) = \Box$, this means that A is modal, that is, A is endowed with a coalgebra structure $A \to \Box A$ for A, and additionally, A behaves in a call-by-value manner; if $\varpi(A) = +$ this means that it is non-modal type which behaves in a call-by-value manner. We define $\Box \odot \Box = \Box$ and $+ \odot \varepsilon = \varepsilon \odot + = +$ for $\varepsilon \in \{\Box, +\}$. It allows composing polarities. Intuitively, if A and B both have a coalgebra for $\Box$, then we can build their product or sum coalgebra; and $\Box A$ is always equipped with the free coalgebra structure.

The introduction rule of $\Box$ states that all types in Γ must be modal: we note $\Gamma^{\Box}$ the restriction of Γ to the types A such that $\varpi(A) = \Box$. A crucial inversion property states that if $\Gamma \nmid \Theta \vdash V : A$ where $\varpi(A) = \Box$, then $\Gamma^{\Box} \nmid \Theta \vdash V : A$, and even more, in a categorical model, the semantics of such a sequent is a coalgebra morphism [CFM16; Mun17]. This only applies to values: there are terms for which this is not true.

The types $\mathbb{1}$, $\otimes$ and $\oplus$ are the usual unit, product and sum types. Additionally, there is a $\mathcal{C}$ control operator in order to make the calculus classical.

Lastly, this calculus is defined in a multiplicative fashion: renamings are explicit in the syntax and contexts are splitted between all the premises. A renaming θ designates a function from the variables of Γ to the ones of Γ' and from the variables of Θ to the ones of Θ'. The syntax is given in fig. 1.

Notice that $\Box t$ is allowed only when t is a value. This makes that $\Box$ is not, stricly speaking, a strong comonad; for example, we do not have $\Box(A \to B) \to \Box A \to \Box B$ in general; but it is still true in a more general sense: we do have $\Box(A \to B) \to \Box A \to \Box(\mathbb{1} \to B)$, and if $x : A \nmid \cdot \vdash V : B; \cdot$ then $\cdot \nmid x : A \vdash \Box V : \Box B; \cdot$. More generally, see [CFM16] for discussions on value restrictions and modality in a call-by-value setting.

$$V, W \quad := \quad x \quad | \quad * \quad | \quad (V, W) \quad | \quad \iota_i\ V \quad | \quad \lambda x.t \quad | \quad \lambda^{\bot} x.t$$

$$t, u, v, w \quad := \quad V \quad | \quad t\ V \quad | \quad t^{\bot}\ V \quad | \quad \mathcal{C}\ t \quad | \quad \text{let } x = t \text{ in } u \quad |$$

$$\text{let } x = t \text{ in}_{\bot} u \quad | \quad \text{let } * = t \text{ in } u \quad | \quad \text{let } * = t \text{ in}_{\bot} u \quad | \quad \text{let } (x, y) = t \text{ in } u \quad |$$

$$\text{let } (x, y) = t \text{ in}_{\bot} u \quad | \quad \text{match } t \text{ with } \{\iota_1\ x.u \mid \iota_2\ y.v\} \quad | \quad \text{match } t \text{ with}_{\bot} \{\iota_1\ x.u \mid \iota_2\ y.v\}$$

(a) Grammar

$$\begin{array}{rcl|c|c|c|c|c} A, B, C & := & \mathbb{1} & A \oplus B & A \otimes B & A \to B & \neg A & \Box A \\ \varpi(A) & := & \Box & \varpi(A) \odot \varpi(B) & \varpi(A) \odot \varpi(B) & + & + & \Box \end{array}$$

(b) Types and polarities

$$\frac{}{\Gamma, x : A, \Gamma' \mid \Theta \vdash x : A} \qquad \frac{}{\Gamma' \mid \Theta, x : A, \Theta' \vdash x : A}$$

$$\frac{\Gamma \mid \Theta \vdash t : \neg\neg A}{\Gamma \mid \Theta \vdash \mathcal{C}\ t : A}$$

$$\frac{\Gamma \mid \Theta \vdash t : A \qquad \Gamma', x : A \mid \Theta' \vdash u : B}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash \text{let } x = t \text{ in } u : B} \qquad \frac{\Gamma \mid \Theta \vdash t : A \qquad \Gamma', x : A \mid \Theta' \vdash u : \bot}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash \text{let } x = t \text{ in}_{\bot} u : \bot}$$

$$\frac{}{\Gamma \mid \Theta \vdash * : \mathbb{1}} \qquad \frac{\Gamma \mid \Theta \vdash V : A \qquad \Gamma' \mid \Theta' \vdash W : B}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash (V, W) : A \otimes B} \qquad \frac{\Gamma \mid \Theta \vdash V : A_i}{\Gamma \mid \Theta \vdash \iota_i\ V : A_1 \oplus A_2}$$

$$\frac{\Gamma^{\Box} \mid \Theta \vdash V : A}{\Gamma^{\Box} \mid \Theta \vdash \Box V : \Box A} \qquad \frac{\Gamma, x : A \mid \Theta \vdash t : B}{\Gamma \mid \Theta \vdash \lambda x.t : A \to B} \qquad \frac{\Gamma, x : A \mid \Theta \vdash t : \bot}{\Gamma \mid \Theta \vdash \lambda^{\bot} x.t : \neg A}$$

$$\frac{\Gamma \mid \Theta \vdash t : \mathbb{1} \qquad \Gamma', x : \mathbb{1} \mid \Theta' \vdash u : A}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash \text{let } * = t \text{ in } u : A} \qquad \frac{\Gamma \mid \Theta \vdash t : \mathbb{1} \qquad \Gamma', x : \mathbb{1} \mid \Theta' \vdash u : \bot}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash \text{let } * = t \text{ in}_{\bot} u : \bot}$$

$$\frac{\Gamma \mid \Theta \vdash t : A \otimes B \qquad \Gamma', x : A, y : B \mid \Theta' \vdash u : C}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash \text{let } (x, y) = t \text{ in } u : C} \qquad \frac{\Gamma \mid \Theta \vdash t : A \otimes B \qquad \Gamma', x : A, y : B \mid \Theta' \vdash u : \bot}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash \text{let } (x, y) = t \text{ in}_{\bot} u : \bot}$$

$$\frac{\Gamma \mid \Theta \vdash t : A \oplus B \qquad \Gamma', x : A \mid \Theta' \vdash u : C \qquad \Gamma', y : B \mid \Theta' \vdash v : C}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash \text{match } t \text{ with } \{\iota_1\ x.u \mid \iota_2\ y.v\} : C}$$

$$\frac{\Gamma \mid \Theta \vdash t : A \oplus B \qquad \Gamma', x : A \mid \Theta' \vdash u : \bot \qquad \Gamma', y : B \mid \Theta' \vdash v : \bot}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash \text{match } t \text{ with}_{\bot} \{\iota_1\ x.u \mid \iota_2\ y.v\} : \bot}$$

$$\frac{\Gamma \mid \Theta \vdash t : \Box A \qquad \Gamma' \mid \Theta', x : A \vdash u : B}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash \text{let } \Box x = t \text{ in } u : B} \qquad \frac{\Gamma \mid \Theta \vdash t : \Box A \qquad \Gamma' \mid \Theta', x : A \vdash u : \bot}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash \text{let } \Box x = t \text{ in}_{\bot} u : \bot}$$

$$\frac{\Gamma \mid \Theta \vdash t : A \to B \qquad \Gamma' \mid \Theta' \vdash V : A}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash t\ V : B} \qquad \frac{\Gamma \mid \Theta \vdash t : \neg A \qquad \Gamma' \mid \Theta' \vdash V : A}{\Gamma, \Gamma' \mid \Theta, \Theta' \vdash t_{\bot}\ V : \bot}$$

$$\frac{\Gamma \mid \Theta \vdash t : A}{\Gamma' \mid \Theta' \vdash \theta(t) : A}\ \theta \text{ renaming} \qquad \frac{\Gamma \mid \Theta \vdash t : \bot}{\Gamma' \mid \Theta' \vdash \theta(t) : \bot}\ \theta \text{ renaming}$$

(c) Typing rules

$$\text{Reduction contexts for } \mathcal{C}: \quad F[\_] \quad := \quad \text{let } x = [\_] \text{ in } t \quad | \quad \text{let } * = [\_] \text{ in } t \quad |$$

$$\text{let } (x, y) = [\_] \text{ in } t \quad | \quad \text{match } [\_] \text{ with } \{\iota_1\ x.t \mid \iota_2\ y.u\} \quad | \quad [\_]\ V \quad | \quad \mathcal{C}\ [\_]$$

$$\begin{array}{rcl@{\qquad}rcl} (\lambda x.t)\ V & \to & t[x := V] & \text{let } (x, y) = (V, W) \text{ in } t & \to & t[x := V, y := W] \\ \text{let } x = V \text{ in } t & \to & t[x := V] & \text{match } \iota_i\ V \text{ with } \{\iota_1\ x_1.t_1 \mid \iota_2\ x_2.t_2\} & \to & t_i[x_i := V] \\ \text{let } * = * \text{ in } t & \to & t & \text{let } \Box x = \Box V \text{ in } t & \to & t[x := V] \end{array}$$

$$\begin{array}{rcl} F[\mathcal{C}\ (\lambda^{\bot} k.t)] & \to & \mathcal{C}\ (\lambda^{\bot} j.t[k := \lambda^{\bot} x.\, \text{let } y = F[x] \text{ in}_{\bot} j^{\bot}\ y]) \\ \mathcal{C}\ (\lambda^{\bot} k.k\ V) & \to & V \text{ (only if } k \text{ is not free in } V, \text{ and the redex is at the toplevel.)} \end{array}$$

(d) Reductions

Figure 1: The $\lambda_{\Box}\mathcal{C}$-calculus

Contrary to Kavvos, this calculus is classical and in call-by-value; we also changed a bit the definition of the introduction rule of $\Box$. In his work, $\Box V$ is typed with Γ empty; in this version, we allow keeping in Γ types with $\varpi(A) = \Box$. This is valid because the semantics of $\Box V$ in his work is to apply the extend operation of the comonad to the semantics of V, and the semantics of our version would to be first compose with the coalgebra of the types in Γ, then applying extend.

The calculus comes with the usual β-reduction rules. The rules for $\mathcal{C}$ are inspired of the ones of [Con+19], and are variant of the ones of [Fel+87; Gri89]. Reductions are allowed in every subterm, except for one of them which can only be applied at toplevel. We do not write the rules for $\bot$ constructs, as they are the same as their non-$\bot$ counterpart. Note that there is no $\bot$ variant for $\mathcal{C}$. In order to define the reduction for $\mathcal{C}$, we need to define a notion of single-frame non-$\bot$ context. The idea is that if $\mathcal{C}\ (\lambda^{\bot} k.t)$ is inside an elementary context $F[\_]$ which is not of $\bot$ type, the $\mathcal{C}$ can bubble up.

Lastly, let us note $\cong$ for the reflexive, symmetric and transitive closure of $\to$.

2.1. CPS

We can define a CPS translation of this calculus into its fragment that does not use the function type $A \to B$, nor the $\mathcal{C}$ operator. This fragment will be called the $\lambda^{\neg}_{\Box}$-calculus. We add to this fragment the η-reduction for functions (it will be needed later, contrary to other η-rules). This means that $\lambda x.V\ x \to V$ and $\lambda^{\bot} x.V^{\bot}\ x \to V$ if x is not free in V.

First, we translate types. The translation corresponds to the usual call-by-value CPS translation for call-by-value [App07, p. 12], but it optimizes the translation of $\neg A$, as in [Con+19]:

$$\begin{array}{rclcrclcrcl} [\![\mathbb{1}]\!]^{\lambda} & = & \mathbb{1} & & [\![A \otimes B]\!]^{\lambda} & = & [\![A]\!]^{\lambda} \otimes [\![B]\!]^{\lambda} & & [\![A \oplus B]\!]^{\lambda} & = & [\![A]\!]^{\lambda} \oplus [\![B]\!]^{\lambda} \\ [\![\Box A]\!]^{\lambda} & = & \Box\, [\![A]\!]^{\lambda} & & [\![\neg A]\!]^{\lambda} & = & \neg\, [\![A]\!]^{\lambda} & & [\![A \to B]\!]^{\lambda} & = & \neg([\![A]\!]^{\lambda} \otimes \neg\, [\![B]\!]^{\lambda}) \end{array}$$

Note that $\varpi([\![A]\!]^{\lambda}) = \varpi(A)$.

Now we define the translation on terms and values. This translation is higher-order, as in [DF92], to avoid introducing unnecessary redexes. There is a translation from values to values $[\![\bullet]\!]^{\lambda}_{\mathsf{v}}$, and a translation $[\![\bullet]\!]^{\lambda}_{\mathsf{t}}\,(\bullet)$ which takes a term of the source and a value of the target representing the continuation. It returns a term. Lastly, there is a translation $[\![\bullet]\!]^{\lambda}_{\bot}$ for terms of $\bot$ type, which also returns a term. We define $\lambda^{\bot}(x,y).t$ to be a notation for $\lambda^{\bot} z.\,\mathrm{let}\,(x,y) = z\ \mathrm{in}_{\bot}\ t$, and $\lambda^{\bot} *.t$ to be a notation for $\lambda^{\bot} z.\,\mathrm{let}\,* = z\ \mathrm{in}_{\bot}\ t$.

$$\begin{array}{rclcrclcrcl} [\![x]\!]^{\lambda}_{\mathsf{v}} & = & x & & [\![\lambda^{\bot} x.t]\!]^{\lambda}_{\mathsf{v}} & = & \lambda^{\bot} x.\,[\![t]\!]^{\lambda}_{\bot} & & [\![\lambda x.t]\!]^{\lambda}_{\mathsf{v}} & = & \lambda^{\bot}(x,k).\,[\![t]\!]^{\lambda}_{\mathsf{t}}\,(k) \\ [\![\Box V]\!]^{\lambda}_{\mathsf{v}} & = & \Box\, [\![V]\!]^{\lambda}_{\mathsf{v}} & & [\![*]\!]^{\lambda}_{\mathsf{v}} & = & * & & [\![\iota_i\ V]\!]^{\lambda}_{\mathsf{v}} & = & \iota_i\ [\![V]\!]^{\lambda}_{\mathsf{v}} \\ [\![(V,W)]\!]^{\lambda}_{\mathsf{v}} & = & ([\![V]\!]^{\lambda}_{\mathsf{v}}, [\![W]\!]^{\lambda}_{\mathsf{v}}) & & & & & & & & \end{array}$$

$$\begin{array}{rclcrcl} [\![V]\!]^{\lambda}_{\mathsf{t}}\,(M) & = & M^{\bot}\ [\![V]\!]^{\lambda}_{\mathsf{v}} & & [\![\mathcal{C}\ t]\!]^{\lambda}_{\mathsf{t}}\,(M) & = & [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda^{\bot} v.v^{\bot}\ M) \\ [\![\mathrm{let}\ x = t\ \mathrm{in}\ u]\!]^{\lambda}_{\mathsf{t}}\,(M) & = & [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda x^{\bot}.\,[\![u]\!]^{\lambda}_{\mathsf{t}}\,(M)) & & [\![\mathrm{let}\ x = t\ \mathrm{in}_{\bot}\ u]\!]^{\lambda}_{\bot} & = & [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda^{\bot} x.\,[\![u]\!]^{\lambda}_{\bot}) \\ [\![\mathrm{let}\ * = t\ \mathrm{in}\ u]\!]^{\lambda}_{\mathsf{t}}\,(M) & = & [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda^{\bot} *.\,[\![u]\!]^{\lambda}_{\mathsf{t}}\,(M)) & & [\![\mathrm{let}\ * = t\ \mathrm{in}_{\bot}\ u]\!]^{\lambda}_{\bot} & = & [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda^{\bot} *.\,[\![u]\!]^{\lambda}_{\bot}) \\ [\![\mathrm{let}\,(x,y) = t\ \mathrm{in}\ u]\!]^{\lambda}_{\mathsf{t}}\,(M) & = & [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda^{\bot}(x,y).\,[\![u]\!]^{\lambda}_{\mathsf{t}}\,(M)) & & [\![\mathrm{let}\,(x,y) = t\ \mathrm{in}_{\bot}\ u]\!]^{\lambda}_{\bot} & = & [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda^{\bot}(x,y).\,[\![u]\!]^{\lambda}_{\bot}) \\ [\![\mathrm{match}\ t\ \mathrm{with}\,\{\iota_1\ x.u \mid \iota_2\ y.v\}]\!]^{\lambda}_{\mathsf{t}}\,(M) & = & [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda^{\bot} z.\,\mathrm{match}\ z\ \mathrm{with}_{\bot}\,\{\iota_1\ x.\,[\![u]\!]^{\lambda}_{\mathsf{t}}\,(M) \mid \iota_2\ y.\,[\![v]\!]^{\lambda}_{\mathsf{t}}\,(M)\}) \\ [\![\mathrm{match}\ t\ \mathrm{with}_{\bot}\,\{\iota_1\ x.u \mid \iota_2\ y.v\}]\!]^{\lambda}_{\bot} & = & [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda^{\bot} z.\,\mathrm{match}\ z\ \mathrm{with}_{\bot}\,\{\iota_1\ x.\,[\![u]\!]^{\lambda}_{\bot} \mid \iota_2\ y.\,[\![v]\!]^{\lambda}_{\bot}\}) \\ [\![t\ V]\!]^{\lambda}_{\mathsf{t}}\,(M) & = & [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda^{\bot} k.k^{\bot}\,([\![V]\!]^{\lambda}_{\mathsf{v}}, \lambda^{\bot} x.M^{\bot}\ x)) & & [\![t^{\bot}\ V]\!]^{\lambda}_{\bot} & = & [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda^{\bot} k.k^{\bot}\ [\![V]\!]^{\lambda}_{\mathsf{v}}) \end{array}$$

In the end, we get that:

- If $\Gamma \mid \Theta \vdash V : A$ then $[\![\Gamma]\!]^\lambda \mid [\![\Theta]\!]^\lambda \vdash [\![V]\!]^\lambda_{\mathsf{v}} : [\![A]\!]^\lambda$
- If $\Gamma \mid \Theta \vdash t : A$ then $[\![\Gamma]\!]^\lambda, k : \neg [\![A]\!]^\lambda \mid [\![\Theta]\!]^\lambda \vdash [\![t]\!]^\lambda_{\mathsf{t}}(k) : \bot$
- If $\Gamma \mid \Theta \vdash t : \bot$ then $[\![\Gamma]\!]^\lambda \mid [\![\Theta]\!]^\lambda \vdash [\![t]\!]^\lambda_{\bot} : \bot$

Lemma 1. If $V \to V'$ (resp. $t \to t'$ and $M \to M'$ or $t \to t'$) then $[\![V]\!]^\lambda_{\mathsf{v}} \cong \mathrm{CPS}\,[\![V']\!]^\Box\, v$ (resp. $[\![t]\!]^\lambda_{\mathsf{t}}(M) \cong \mathrm{CPS}\,[\![t']\!]^\Box\, tM'$ and $[\![t]\!]^\lambda_{\bot} \cong [\![t']\!]^\lambda_{\bot}$), and we have the same theorem with $\to^*$ instead of $\cong$ if the contracted redex is not $F[\mathcal{C}\ (\lambda^{\bot} k.t)]$.

For the proof, see section A.1.

3. System $\mathbf{L}^{\Box}_{\mathsf{pol}}$

We devised in the previous section a λ-calculus for the classical S4 modal logic, but λ-calculi are representations of natural deduction systems, which is ill-behaved for classical logic in contrast to sequent calculus. Thus, we devised another system, more complete, which is more fitting for sequent calculus: a $\bar{\lambda}\mu\tilde{\mu}$-calculus [CH00]. Here, we rely on a polarised variant, which means it supports both call-by-name and call-by-value. Its formulas are inspired by the system LC of Girard [Gir91] and LK^{η}_{pol} of Danos, Joinet and Schellinx [DJS97], which are polarised system for classical logic, but without term assignment system. A term assignment system for those has been designed by my supervisor in [Mun09], and the system presented here is adapted from it. We do not support the full extent of the connectives (there is no negative pair, no positive negation, no false type, no negative true type).

3.1. Syntax

- Polarities $\varepsilon := + \mid -$
- Values $V, W := x \mid \mu\alpha^{-}.c \mid * \mid (V, W) \mid \iota_i V \mid \Box V \mid \{V\} \mid \mu[x].c \mid \mu(\alpha, \beta).c \mid \mu\{\alpha\}.c$
- Terms $t := V \mid \mu\alpha^{+}.c$
- Stacks : $S, S' := \alpha \mid \tilde{\mu}x^{+}.c \mid \tilde{\mu} * .c \mid \tilde{\mu}(x, y).c \mid \tilde{\mu}(\iota_1\ x.c \mid \iota_2\ y.c') \mid \tilde{\mu}\Box x.c \mid \tilde{\mu}\{x\}.c \mid [V] \mid (S, S') \mid \{S\}$
- Environments : $e := S \mid \tilde{\mu}x^{-}.c$
- Commands : $c, c' := \langle t \mid^{+} S\rangle \mid \langle V \mid^{-} e\rangle$.

There are five syntactic classes in the syntax: values, terms, stacks, environments and commands. Commands correspond to programs where no input nor output is focused, and are introduced by the cut rules. Values and terms represent a focus on the right of the $\vdash$, and stacks and environments represent a focus on its left part. The difference between values and terms, respectively stacks and environments, is that one can substitute variables, respectively covariables only by values, respectively by stacks, not by arbitrary terms or environments. Intuitively, $\langle V \mid^{-} e\rangle$ correspond to the program obtained by placing a value V in some environment e, and $\langle t \mid^{+} S\rangle$ to the program obtained by giving a stack S to some term t.

Constructs starting with μ represent the capture of some continuation or stack, and they can

optionally pattern match on it. Symmetrically, constructs starting by $\tilde{\mu}$ represent the capture of some value by the environment (i.e., a let), and then optionally a pattern matching on it.

For example, a (V, W) represents the pair of two values and $\tilde{\mu}(x, y).c$ represents a program which pattern matches on the value it waits for, assigns its member to x and y and continue with c; $[V]$ represents a stack where we putted the value V at the bottom, and $\mu[x].c$ represents a program that will inspect a stack of this form, retrieve V, assign it to x and continue with c.

We will write $\langle t \mid^{\varepsilon} e\rangle$ or even $\langle t \mid e\rangle$ when it is not necessary to mention the polarity. We will write $\mu\alpha^{\varepsilon}.c$ and $\tilde{\mu}x^{\varepsilon}.c$ as well.

3.2. Reductions

We have β-reductions and η-reductions:[1]

$$\begin{array}{rcc|rcc}
\langle \mu\alpha^{\varepsilon}.c \mid^{\varepsilon} S\rangle & \to_\beta & c[\alpha := S] & \mu\alpha^{\varepsilon}.\langle t \mid^{\varepsilon} \alpha\rangle & \to_\eta & t \\
\langle V \mid^{\varepsilon} \tilde{\mu}x^{\varepsilon}.c\rangle & \to_\beta & c[x := V] & \tilde{\mu}x^{\varepsilon}.\langle x \mid^{\varepsilon} e\rangle & \to_\eta & e \\
\langle * \mid^{+} \tilde{\mu} * .c\rangle & \to_\beta & c & \mu[x].\langle V \mid^{-} [x]\rangle & \to_\eta & V \\
\langle (V, W) \mid^{+} \tilde{\mu}(x, y).c\rangle & \to_\beta & c[x := V, y := W] & \mu(\alpha, \beta).\langle V \mid^{-} (\alpha, \beta)\rangle & \to_\eta & V \\
\langle \iota_i\ V \mid^{+} \tilde{\mu}(\iota_1\ x_1.c_1 \mid \iota_2\ x_2.c_2)\rangle & \to_\beta & c_i[x_i := V] & \mu\{\alpha\}.\langle V \mid^{-} \{\alpha\}\rangle & \to_\eta & V \\
\langle \Box V \mid^{+} \tilde{\mu}\Box x.c\rangle & \to_\beta & c[x := V] & \tilde{\mu} * .\langle * \mid^{+} S\rangle & \to_\eta & S \\
\langle \{V\} \mid^{+} \tilde{\mu}\{x\}.c\rangle & \to_\beta & c[x := V] & \tilde{\mu}(x, y).\langle (x, y) \mid S\rangle & \to_\eta & S \\
\langle \mu[x].c \mid^{-} [V]\rangle & \to_\beta & c[x := V] & \tilde{\mu}(\iota_1\ x.\langle \iota_1\ x \mid^{\varepsilon_1} S\rangle \mid \iota_2\ y.\langle \iota_2\ y \mid^{\varepsilon_2} S\rangle) & \to_\eta & S \\
\langle \mu(\alpha, \beta).c \mid^{-} (e, f)\rangle & \to_\beta & c[\alpha := e, \beta := f] & \tilde{\mu}\Box x.\langle \Box x \mid^{\Box} S\rangle & \to_\eta & S \\
\langle \mu\{\alpha\}.c \mid^{-} \{S\}\rangle & \to_\beta & c[\alpha := S] & \tilde{\mu}\{x\}.\langle \{x\} \mid^{+} S\rangle & \to_\eta & S
\end{array}$$

We oriented these rules as reductions instead of expansions because we want to do untyped reductions. In this direction, they have the property of subject reduction. They are not confluent, because for example, one has $\tilde{\mu}(x, y).c[z := (x, y)] \leftarrow_\beta \tilde{\mu}(x, y).\langle (x, y) \mid^{+} \tilde{\mu}z^{+}.c\rangle \to_\eta \tilde{\mu}z^{+}.c$, which cannot always be joined. However, following [Mun17], we will restrict the reduction $\to_{\beta\eta}$ to the reduction $\to$ which allows all β-reductions, but restrict the η-rules and keep only the one of $\mu\alpha^{\varepsilon}$ and $\tilde{\mu}x^{\varepsilon}$. The reflexive, symmetric and transitive closure of $\to$ will also be noted $\cong$.

Lemma 2 (Confluence)**.** The reduction $\to$ is confluent, as it is a weakly orthogonal rewrite system [OR94].

3.3. Typing

$$\begin{array}{lcccccccccccc}
\text{Types} & : & A, B & = & \mathbb{1} & A \otimes B & A \oplus B & \Box A & \Downarrow A & \neg A & A \parr B & \Uparrow A \\
\varpi & : & \varpi(A) & = & \Box & \varpi(A) \odot \varpi(B) & \varpi(A) \odot \varpi(B) & \Box & \varpi(A)^{\star} & - & - & -
\end{array}$$

where $\varpi(A)^{\star} = \begin{cases} \Box & \text{if } \varpi(A) = \Box \\ + & \text{else} \end{cases}$.

To the connectives of linear logic without the exponentials, we added a connective $\Box$, representing a comonad. However, the system is not linear: that is way the comonad is $\Box$ and not !. Thus, even though $A \oplus B$ and $A \parr B$ are logically equivalent, they are not isomorphic and their reduction rules differ. Types have a polarity, which is positive or negative. Additionally,

[1] x, y, α and β must not be free in V, S, t or e.

positive types can be modal: that means that they also have a coalgebra structure for the $\Box$. We note $\varpi(A) = -$ for negative types, $\varpi(A) = +$ for positive types which are not modal, and $\varpi(A) = \Box$ for modal types. Positive types correspond to call-by-value types, and negative ones to call-by-name types. Like before, we define $\Box \odot \Box = \Box$ and $\varepsilon \odot \varepsilon' = +$ if $\varepsilon, \varepsilon' \in \{\Box, +, -\}$ and $(\varepsilon, \varepsilon') \neq (\Box, \Box)$.

We define the type of booleans $\mathbb{B}$ to be the type $\mathbb{1} \oplus \mathbb{1}$, with its two constructors true $:= \iota_1 *$ and false $:= \iota_2 *$.

One can see that there is no function types in this system. The type $A \to B$ can be macro-defined as $\neg A \parr B$, with $\parr$ representing the negative disjunction ; values of type $A \to B$ are introduced by $\mu(a \cdot \beta).c := \mu(\alpha, \beta).\langle \mu[a].c \mid^- \alpha \rangle$ — in call-by-name, $\lambda x.t$ will be defined as $\mu(x \cdot \beta).\langle [\![t]\!] \mid^- \beta \rangle$: the idea that a value of type $A \to B$ captures a value of type A and a continuation of type B: the returned value must be sent to this continuation. Stacks of type $A \to B$ are introduced by $V \cdot S := ([V], S)$. The idea is that a stack of type $A \to B$ consists of a value of type V on top of a stack of type B, which represent the stack on which the result of the function will run: for example, $t\ u$ will be translated as $\mu\beta^-.\langle [\![t]\!] \mid^- [\![u]\!] \cdot \beta \rangle$ ($[\![u]\!]$ is a value because in call-by-name, every term is translated as a value of negative type).

We added explicit polarity shifts $\Downarrow$ and $\Uparrow$. It is possible to express them inside the system because $\Downarrow A \simeq A \otimes \mathbb{1}$ and $\Uparrow A \simeq \mathbb{1} \to A$, but having them as explicit connectives gives us a better CPS translation: we can simply erase them.

3.3.1. Rules

We consider the typing rules given in fig. 2. There are five typing judgments: $\Gamma \nmid \Theta \vdash V : A; \Delta$, $\Gamma \nmid \Theta \vdash t : A \mid \Delta$, $\Gamma \nmid \Theta; S : A \vdash \Delta$, $\Gamma \nmid \Theta \mid e : A \vdash \Delta$ and $c : (\Gamma \nmid \Theta \vdash \Delta)$.

The four firsts read as follows: "in the context $\Gamma \nmid \Theta \vdash \Delta$, V, respectively t, S or e is a value, respectively a term, a stack or an environment of type A". The last one is read: "c is a command in the context $\Gamma \nmid \Theta \vdash \Delta$".

Sequents are of the shape $\Gamma \nmid \Theta \vdash \Delta$ with three contexts Γ, Θ and Δ, where Γ is the usual context, Θ is the modal context, as seen in the previous section, and Δ is a co-context, consisting of co-variables $\alpha : A$, representing continuations. Indeed, following the Curry-Howard correspondence for classical logic, we know that formulas on the right-hand side of the $\vdash$ corresponds to continuations variables. This is to be compared with the usual λ-calculus because it is the syntax for intuitionistic logic where there is always at most one formula on the right. Lastly, sequents are also optionally composed of a distinguished zone (except for commands judgments), which can be left or right, and contains at most one formula. Intuitively, the formula in this zone is the one we are working on.

When writing Γ, Γ', Θ, Θ' or Δ, Δ', we suppose that Γ and Γ', Θ and Θ' and Δ and Δ' are disjoint.

$\Gamma^\Box$ is the restriction of Γ to the types A such that $\varpi(A) = \Box$.

We define renamings $\theta \in \mathfrak{R}(\Gamma \nmid \Theta \vdash \Delta \Rightarrow \Gamma' \nmid \Theta' \vdash \Delta')$: θ is a function between variables and covariables such that $x : A \in \Gamma$ then $\theta(x) : A \in \Gamma'$, if $x : A \in \Theta$ then $\theta(x) : A \in \Theta'$ and if $\alpha : A \in \Delta$ then $\theta(\alpha) : A \in \Delta'$.

Lemma 3 (Restriction to free variables)**.** By induction, one can show that if $\Gamma \nmid \Theta \vdash V : A; \Delta$,

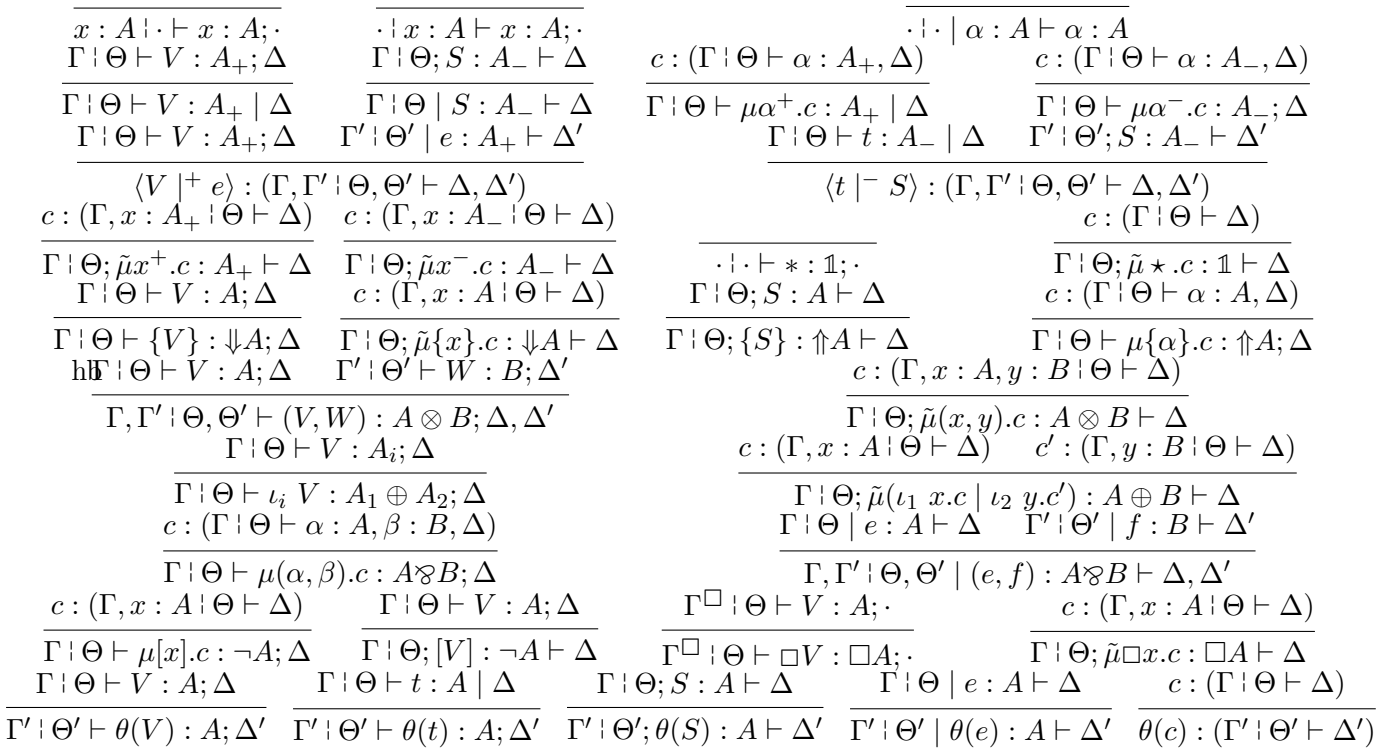


Figure 2: Typing rules of the system $\mathbf{L}^{\Box}_{\mathsf{pol}}$

or $c : (\Gamma \mid \Theta \vdash \Delta)$ or $\Gamma \mid \Theta \mid e : A \vdash \Delta$, etc., one can restrict the context $\Gamma \mid \Theta \vdash \Delta$ to the free variables of V, c, e, etc.

Indeed, we can always restrict θ to be a surjective function in the renaming rule.

Lemma 4 (Restriction of modal values)**.** We can also show that if $\Gamma \mid \Theta \vdash V : A; \Delta$ where $\varpi(A) = \Box$, then $\Gamma^{\Box} \mid \Theta \vdash V : A; \cdot$.

For the proof, it is enough to show that free variables of a modal value are all of modal type, or belongs to Θ. Clearly, this is true. We need V to be a value: this is not true, for example, if we consider the $z : A \otimes B \mid \cdot \vdash \mu\alpha^{+}.\langle z \mid^{+} \tilde{\mu}(x, y).\langle x \mid^{+} \alpha\rangle\rangle : A; \cdot$ which retrieves the first projection of z, where A is modal but B is not, because then, $A \otimes B$ is not modal, so $\Gamma^{\Box}$ is empty.

3.3.2. Substitution

Definition 1 (Typed substitution)**.** $\sigma : (\Gamma \mid \Theta \vdash \Delta) \Rightarrow (\Gamma' \mid \Theta' \vdash \Delta')$ is a typed substitution if:

- for every $x : A \in \Gamma$, $\sigma(x)$ is a value such that $\Gamma' \mid \Theta' \vdash \sigma(x) : A; \Delta'$;
- for every $x : A \in \Theta$, $\sigma(x)$ is a value such that $\Gamma'^{\Box} \mid \Theta' \vdash \sigma(x) : A; \cdot$ (what we really want is that $\Gamma' \mid \Theta' \vdash \Box\sigma(x) : \Box A; \Delta'$) ;
- for every $\alpha : A \in \Delta$, $\sigma(\alpha)$ is a stack such that $\Gamma \mid \Theta; \sigma(\alpha) : A \vdash \Delta$.

Lemma 5 (Compatibility of substitutions with typing)**.** If $\sigma : (\Gamma \mid \Theta \vdash \Delta) \Rightarrow (\Gamma' \mid \Theta' \vdash \Delta')$ then:

- $\Gamma \mid \Theta \vdash V : A; \Delta \implies \Gamma' \mid \Theta' \vdash \sigma(V) : A; \Delta'$
- $\Gamma \mid \Theta \vdash t : A \mid \Delta \implies \Gamma' \mid \Theta' \vdash \sigma(t) : A \mid \Delta'$
- $\Gamma \mid \Theta; S : A \vdash \Delta \implies \Gamma' \mid \Theta'; \sigma(S) : A \vdash \Delta'$
- $\Gamma \mid \Theta \mid e : A \vdash \Delta \implies \Gamma' \mid \Theta' \mid \sigma(e) : A \vdash \Delta'$
- $c : (\Gamma \mid \Theta \vdash \Delta) \implies \sigma(c) : (\Gamma' \mid \Theta' \vdash \Delta')$

For the proof, see the section A.2.

3.3.3. Inversion

The renaming rules can be inserted everywhere, making induction on terms difficult. But successive renaming rules can be composed, and one can always insert a dummy renaming rule with the identity as renaming. Therefore, we can show the following inversion lemma:

Lemma 6 (Inversion)**.** For every proof of $\Gamma \mid \Theta \vdash (V, W) : A \otimes B; \Delta$ there are sequents $(\Gamma_1 \mid \Theta_1 \vdash \Delta_1)$ and $(\Gamma_2 \mid \Theta_2 \vdash \Delta_2)$, as well as a renaming $\theta \in \mathfrak{R}(\Gamma_1, \Gamma_2 \mid \Theta_1, \Theta_2 \vdash \Delta_1, \Delta_2 \Rightarrow \Gamma \mid \Theta \vdash \Delta)$ and values V' and W' such that $\theta((V', W')) = (V, W)$ and the proof can be written as follows:

$$\dfrac{\dfrac{\Gamma_1 \mid \Theta_1 \vdash V' : A; \Delta_1 \qquad \Gamma_2 \mid \Theta_2 \vdash W' : B; \Delta_2}{\Gamma_1, \Gamma_2 \mid \Theta_1, \Theta_2 \vdash (V', W') : A \otimes B; \Delta_1, \Delta_2}}{\Gamma \mid \Theta \vdash (V, W) : A \otimes B; \Delta}$$

One can show similar inversion lemmas for other connectives. For example, for variables there is always a derivation of the following form:

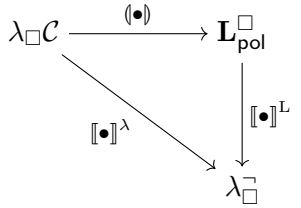
$$\dfrac{x : A \mid \cdot \vdash x : A; \cdot}{\Gamma, x : A, \Gamma' \mid \Theta \vdash x : A; \Delta} \quad \text{or} \quad \dfrac{\cdot \mid x : A \vdash x : A; \cdot}{\Gamma \mid \Theta, x : A, \Theta' \vdash x : A; \Delta}$$

3.3.4. Subject reduction

Equipped with the inversion lemma, one can show that if $c \to_{\beta\eta} c'$ et $c : (\Gamma \mid \Theta \vdash \Delta)$, then $c' : (\Gamma \mid \Theta \vdash \Delta)$, and similar results for values, terms, stacks and environments.

4. Refinement of the $\lambda_{\Box}\mathcal{C}$-calculus

In order to show that the system $\mathbf{L}^{\Box}_{\mathsf{pol}}$ indeed refines our previous λ-calculus, we can show that there is a translation $(\!|\bullet|\!)$ from the $\lambda_{\Box}\mathcal{C}$-calculus to this system which exhibits the $\lambda_{\Box}\mathcal{C}$-calculus constructions as macros in the $\mathbf{L}^{\Box}_{\mathsf{pol}}$ system, and a CPS translation $[\![\bullet]\!]^{\mathrm{L}}$ from this system to the target $\lambda^{\neg}_{\Box}$-calculus, factorizing the previous CPS translation, that is, the following diagram commutes:



From this diagram, we must understand than the system $\mathbf{L}^\Box_{\mathsf{pol}}$ is more fine-grained than the $\lambda_\Box\mathcal{C}$-calculus, because everything that can be expressed in the $\lambda_\Box\mathcal{C}$-calculus can also be macro-expressed in the system $\mathbf{L}^\Box_{\mathsf{pol}}$, and every terms which are equalized in the translation into the system $\mathbf{L}^\Box_{\mathsf{pol}}$ are also equalized through the CPS translation; yet because the translation targets the $\lambda^\neg_\Box$-calculus, we know that it this system is not "too powerful", i.e. inconsistent, because the $\lambda^\neg_\Box$ does not prove $\bot$.

Lemma 7. For all types A, $[\![A]\!]^\lambda = [\![(\!|A|\!)]\!]^+$.

For all values V, terms t of non-$\bot$ type and terms u of $\bot$ type, and for all stacks S, we have:

- $[\![(\!|V|\!)_{\mathsf{v}}]\!]^{\mathrm{L}}_{\mathsf{v}} \to^* [\![V]\!]^\lambda_{\mathsf{v}}$
- $[\![(\!|t|\!)_{\mathsf{t}}(S)]\!]^{\mathrm{L}}_{\mathsf{c}} \to^* [\![t]\!]^\lambda_{\mathsf{t}}\,([\![S]\!]^{\mathrm{L}}_{\mathsf{s}})$
- $[\![(\!|u|\!)_\bot]\!]^{\mathrm{L}}_{\mathsf{c}} \to^* [\![u]\!]^\lambda_\bot$

For the proof, see section A.4 after having read this section.

4.1. The translation $(\!|\bullet|\!)$

The translation from $\lambda_\Box\mathcal{C}$ to $\mathbf{L}^\Box_{\mathsf{pol}}$ is quite straightforward. Once again, it is higher-order in order to avoid administrative redexes, which can be erased immediately. The translation depends on whether we have a value, a term or a term that returns $\bot$. The translation of $\mathcal{C}\ t$ simply implement $\neg\neg A \to A$ in the system $\mathbf{L}^\Box_{\mathsf{pol}}$.

Since the $\lambda_\Box\mathcal{C}$-calculus is in call-by-value, $(\!|T|\!)$ has to be a positive type for every T. Thus, we translated $A \to B$ as $\Downarrow(\neg A \parr B)$ and $\neg A$ as $\Downarrow\neg A$.

$$\begin{array}{llllllll}
(\!|\mathbb{1}|\!) & = & \mathbb{1} & (\!|A \otimes B|\!) & = & (\!|A|\!) \otimes (\!|B|\!) & (\!|A \oplus B|\!) & = & (\!|A|\!) \oplus (\!|B|\!) \\
(\!|\Box A|\!) & = & \Box(\!|A|\!) & (\!|A \to B|\!) & = & \Downarrow(\neg(\!|A|\!) \parr (\!|B|\!)) & (\!|\neg A|\!) & = & \Downarrow(\neg(\!|A|\!))
\end{array}$$

$$(\!|x|\!)_{\mathsf{v}} = x \qquad (\!|*|\!)_{\mathsf{v}} = * \qquad (\!|(V,W)|\!)_{\mathsf{v}} = ((\!|V|\!)_{\mathsf{v}}, (\!|W|\!)_{\mathsf{v}}) \qquad (\!|\iota_i\ V|\!)_{\mathsf{v}} = \iota_i\ (\!|V|\!)_{\mathsf{v}}$$
$$(\!|\lambda a.t|\!)_{\mathsf{v}} = \{\mu(\alpha,\beta).\langle \mu[a].(\!|t|\!)_{\mathsf{t}}(\beta) \mid^- \alpha\rangle\} \qquad (\!|\lambda^\bot a.t|\!)_{\mathsf{v}} = \{\mu[a].(\!|t|\!)_\bot\}$$

$$\begin{array}{lcl}
(\!|V|\!)_{\mathsf{t}}(S) & = & \langle (\!|V|\!)_{\mathsf{v}} \mid^+ S\rangle \\
(\!|\mathcal{C}\ t|\!)_{\mathsf{t}}(S) & = & (\!|t|\!)_{\mathsf{t}}(\tilde{\mu}\{x\}.\langle x \mid^- [\{\mu[y].\langle y \mid^+ S\rangle\}]\rangle) \\
(\!|t\ V|\!)_{\mathsf{t}}(S) & = & (\!|t|\!)_{\mathsf{t}}(\tilde{\mu}\{f\}.\langle f \mid^- ([(\!|V|\!)_{\mathsf{v}}], S)\rangle) \\
(\!|t^\bot\ V|\!)_\bot & = & (\!|t|\!)_{\mathsf{t}}(\tilde{\mu}\{f\}.\langle f \mid^- [(\!|V|\!)_{\mathsf{v}}]\rangle) \\
(\!|\text{let } x = t \text{ in } u|\!)_{\mathsf{t}}(S) & = & (\!|t|\!)_{\mathsf{t}}(\tilde{\mu}x^+.(\!|u|\!)_{\mathsf{t}}(S)) \\
(\!|\text{let } x = t \text{ in}_\bot\ u|\!)_\bot & = & (\!|t|\!)_{\mathsf{t}}(\tilde{\mu}x^+.(\!|u|\!)_\bot)
\end{array}$$

We have that:

- $\Gamma \mid \Theta \vdash V : A \implies (\!|\Gamma|\!) \mid (\!|\Theta|\!) \vdash (\!|V|\!)_{\mathsf{v}} : (\!|A|\!)$;
- $\Gamma \mid \Theta \vdash t : A \implies (\!|t|\!)_{\mathsf{t}}(\alpha) : ((\!|\Gamma|\!) \mid (\!|\Theta|\!) \vdash \alpha : (\!|A|\!))$;
- $\Gamma \mid \Theta \vdash t : \bot \implies (\!|t|\!)_{\bot} : ((\!|\Gamma|\!) \mid (\!|\Theta|\!) \vdash \cdot)$.

Lemma 8. If $V \to V'$, respectively $t \to t'$ and $S \to S'$ or $t \to t'$ then $(\!|V|\!)_{\mathsf{v}} \cong (\!|V'|\!)_{\mathsf{v}}$, respectively $(\!|t|\!)_{\mathsf{t}}(S) \cong (\!|t'|\!)_{\mathsf{t}}(S')$ or $(\!|t|\!)_{\bot} \cong (\!|t'|\!)_{\bot}$, and we have the same theorem with $\to^*$ instead of $\cong$ if the contracted redex is not $F[\mathcal{C}\ (\lambda^{\bot} k.t)]$.

For the proof, see section A.3.

4.2. CPS translation

We will now devise the CPS translation for this calculus. First of all, we have two translations for types; one gives the types of values, $[\![\bullet]\!]^+$, the other one gives the types of stacks, $[\![\bullet]\!]^-$.

$$\begin{array}{llllllllll} [\![A_-]\!]^+ & = & \neg [\![A_-]\!]^- & [\![\Box A]\!]^+ & = & \Box [\![A]\!]^+ & [\![A \otimes B]\!]^+ & = & [\![A]\!]^+ \otimes [\![B]\!]^+ & [\![\Downarrow A]\!]^+ & = & [\![A]\!]^+ \\ & & [\![\mathbb{1}]\!]^+ = \mathbb{1} & & & & [\![A \oplus B]\!]^+ = [\![A]\!]^+ \oplus [\![B]\!]^+ & & & & & \\ [\![A_+]\!]^- & = & \neg [\![A_+]\!] & [\![\neg A]\!]^- & = & [\![A]\!]^+ & [\![A \parr B]\!]^- & = & [\![A]\!]^- \otimes [\![B]\!]^- & [\![\Uparrow A]\!]^- & = & [\![A]\!]^- \end{array}$$

In particular, for B positive we have $[\![\neg A \parr B]\!]^+ = \neg([\![A]\!]^+ \otimes \neg [\![B]\!]^+)$, which corresponds to the usual call-by-value translation of arrows [App07, p. 12].

Then, we have the translations of values and stacks — $[\![\bullet]\!]^{\mathrm{L}}_{\mathsf{v}}$ and $[\![\bullet]\!]^{\mathrm{L}}_{\mathsf{s}}$ — which returns values, and the translations of terms and environments — $[\![\bullet]\!]^{\mathrm{L}}_{\mathsf{t}}(\bullet)$ and $[\![\bullet]\!]^{\mathrm{L}}_{\mathsf{e}}(\bullet)$, which also take a continuation (which is a value) and returns a term. Finally, commands are translated as terms with $[\![\bullet]\!]^{\mathrm{L}}_{\mathsf{c}}$. We have that:

- $\Gamma \mid \Theta \vdash V : A; \Delta \implies [\![\Gamma]\!]^+, [\![\Delta]\!]^- \mid \Theta \vdash [\![V]\!]^{\mathrm{L}}_{\mathsf{v}} : [\![A]\!]^+$
- $\Gamma \mid \Theta \vdash t : A \mid \Delta \implies [\![\Gamma]\!]^+, [\![\Delta]\!]^-, k : [\![A]\!]^- \mid \Theta \vdash [\![t]\!]^{\mathrm{L}}_{\mathsf{t}}(k) : \bot$
- $\Gamma \mid \Theta; S : A \vdash \Delta \implies [\![\Gamma]\!]^+, [\![\Delta]\!]^- \mid \Theta \vdash [\![S]\!]^{\mathrm{L}}_{\mathsf{s}} : [\![A]\!]^-$
- $\Gamma \mid \Theta \mid e : A \vdash \Delta \implies [\![\Gamma]\!]^+, [\![\Delta]\!]^-, k : [\![A]\!]^+ \mid \Theta \vdash [\![e]\!]^{\mathrm{L}}_{\mathsf{e}}(k) : \bot$
- $c : (\Gamma \mid \Theta \vdash \Delta) \implies [\![\Gamma]\!]^+, [\![\Delta]\!]^- \mid \Theta \vdash [\![c]\!]^{\mathrm{L}}_{\mathsf{c}} : \bot$

$[\![x]\!]^{\mathrm{L}}_{\mathsf{v}}$	$=$	x	$[\![\tilde{\mu} x^+.c]\!]^{\mathrm{L}}_{\mathsf{s}}$	$=$	$\lambda^{\bot} x.\, [\![c]\!]^{\mathrm{L}}_{\mathsf{c}}$
$[\![*]\!]^{\mathrm{L}}_{\mathsf{v}}$	$=$	$*$	$[\![\tilde{\mu} * .c]\!]^{\mathrm{L}}_{\mathsf{s}}$	$=$	$\lambda^{\bot} * .\, [\![c]\!]^{\mathrm{L}}_{\mathsf{c}}$
$[\![(V,W)]\!]^{\mathrm{L}}_{\mathsf{v}}$	$=$	$([\![V]\!]^{\mathrm{L}}_{\mathsf{v}}, [\![W]\!]^{\mathrm{L}}_{\mathsf{v}})$	$[\![\tilde{\mu}(x,y).c]\!]^{\mathrm{L}}_{\mathsf{s}}$	$=$	$\lambda^{\bot}(x,y).\, [\![c]\!]^{\mathrm{L}}_{\mathsf{c}}$
$[\![\iota_i\ V]\!]^{\mathrm{L}}_{\mathsf{v}}$	$=$	$\iota_i\ [\![V]\!]^{\mathrm{L}}_{\mathsf{v}}$	$[\![\tilde{\mu}(\iota_1\ x.c \mid \iota_2\ y.c')]\!]^{\mathrm{L}}_{\mathsf{s}}$	$=$	$\lambda^{\bot} z.\, \mathrm{match}\ z\ \mathrm{with}_{\bot}\ \{\iota_1\ x.\, [\![c]\!]^{\mathrm{L}}_{\mathsf{c}} \mid \iota_2\ y.\, [\![c']\!]^{\mathrm{L}}_{\mathsf{c}}\}$
$[\![\Box V]\!]^{\mathrm{L}}_{\mathsf{v}}$	$=$	$\Box\, [\![V]\!]^{\mathrm{L}}_{\mathsf{v}}$	$[\![\tilde{\mu} \Box x.c]\!]^{\mathrm{L}}_{\mathsf{s}}$	$=$	$\lambda^{\bot} \Box x.\, [\![c]\!]^{\mathrm{L}}_{\mathsf{c}}$
$[\![\{V\}]\!]^{\mathrm{L}}_{\mathsf{v}}$	$=$	$[\![V]\!]^{\mathrm{L}}_{\mathsf{v}}$	$[\![\tilde{\mu}\{x\}.c]\!]^{\mathrm{L}}_{\mathsf{s}}$	$=$	$\lambda^{\bot} x.\, [\![c]\!]^{\mathrm{L}}_{\mathsf{c}}$
$[\![\alpha]\!]^{\mathrm{L}}_{\mathsf{s}}$	$=$	α	$[\![\mu\alpha^-.c]\!]^{\mathrm{L}}_{\mathsf{v}}$	$=$	$\lambda^{\bot} \alpha.\, [\![c]\!]^{\mathrm{L}}_{\mathsf{c}}$
$[\![V]\!]^{\mathrm{L}}_{\mathsf{s}}$	$=$	$[\![V]\!]^{\mathrm{L}}_{\mathsf{v}}$	$[\![\mu[x].c]\!]^{\mathrm{L}}_{\mathsf{v}}$	$=$	$\lambda^{\bot} x.\, [\![c]\!]^{\mathrm{L}}_{\mathsf{c}}$
$[\![(S,T)]\!]^{\mathrm{L}}_{\mathsf{s}}$	$=$	$([\![S]\!]^{\mathrm{L}}_{\mathsf{s}}, [\![T]\!]^{\mathrm{L}}_{\mathsf{s}})$	$[\![\mu(\alpha,\beta).c]\!]^{\mathrm{L}}_{\mathsf{v}}$	$=$	$\lambda^{\bot}(\alpha,\beta).\, [\![c]\!]^{\mathrm{L}}_{\mathsf{c}}$
$[\![\{S\}]\!]^{\mathrm{L}}_{\mathsf{s}}$	$=$	$[\![S]\!]^{\mathrm{L}}_{\mathsf{s}}$	$[\![\mu\{\alpha\}.c]\!]^{\mathrm{L}}_{\mathsf{v}}$	$=$	$\lambda^{\bot} \alpha.\, [\![c]\!]^{\mathrm{L}}_{\mathsf{c}}$

$[\![V]\!]^{\mathrm{L}}_{\mathsf{t}}(M)$	$=$	$M^{\bot}\ [\![V]\!]^{\mathrm{L}}_{\mathsf{v}}$	$[\![\mu\alpha^+.c]\!]^{\mathrm{L}}_{+}(M)$	$=$	$\mathrm{let}\ \alpha = M\ \mathrm{in}_{\bot}\ [\![c]\!]^{\mathrm{L}}_{\mathsf{c}}$
$[\![S]\!]^{\mathrm{L}}_{-}(M)$	$=$	$M^{\bot}\ [\![S]\!]^{\mathrm{L}}_{\mathsf{s}}$	$[\![\tilde{\mu} x^-.c]\!]^{\mathrm{L}}_{-}(M)$	$=$	$\mathrm{let}\ x = M\ \mathrm{in}_{\bot}\ [\![c]\!]^{\mathrm{L}}_{\mathsf{c}}$
$[\![\langle t \mid^+ S \rangle]\!]^{\mathrm{L}}_{\mathsf{c}}$	$=$	$[\![t]\!]^{\mathrm{L}}_{+}([\![S]\!]^{\mathrm{L}}_{\mathsf{s}})$	$[\![\langle V \mid^- e \rangle]\!]^{\mathrm{L}}_{\mathsf{c}}$	$=$	$[\![e]\!]^{\mathrm{L}}_{-}([\![V]\!]^{\mathrm{L}}_{\mathsf{v}})$

Lemma 9 ($[\![\bullet]\!]^{\mathrm{L}}$ is a simulation)**.** If $t \to t'$, respectively $M \to M'$, $V \to V'$, $e \to e'$, $M \to M'$, $S \to S'$ or $c \to c'$ with t, V, e, S and c well-typed, and if M is a value, then $[\![t]\!]^{\mathrm{L}}_{\mathsf{t}}(M) \to [\![t']\!]^{\mathrm{L}}_{\mathsf{t}}(M)$, respectively $[\![t]\!]^{\mathrm{L}}_{\mathsf{t}}(M) \to [\![t]\!]^{\mathrm{L}}_{\mathsf{t}}(M')$, $[\![V]\!]^{\mathrm{L}}_{\mathsf{v}} \to [\![V']\!]^{\mathrm{L}}_{\mathsf{v}}$, etc.

However, note that this translation is not compatible with other η-rules as the one of $\tilde{\mu}(x, y)$, because we did not add other η-rules in $\lambda_{\Box}\mathcal{C}$. Nevertheless, it would work if they were added.

Proof. For β-reduction, it is more or less obvious. For the η-reduction of $\mu\alpha^+$ and $\tilde{\mu}x^-$, too. For $\mu\alpha^-$ and $\tilde{\mu}x^+$, we need to use the η-rule for functions: $[\![\mu\alpha^-.\langle V \mid^- \alpha\rangle]\!]^{\mathrm{L}}_{\mathsf{v}} = \lambda^{\perp}\alpha.([\![V]\!]^{\mathrm{L}}_{\mathsf{v}})^{\perp}\ \alpha \to [\![V]\!]^{\mathrm{L}}_{\mathsf{v}}$ because $[\![V]\!]^{\mathrm{L}}_{\mathsf{v}}$ is a value and α is not free V. □

5. Normalization

We now would now like to prove the consistency of the system $\mathbf{L}^{\Box}_{\mathsf{pol}}$. As said in the previous section, thanks to the CPS translation, we know that it does not prove $\perp$. Moreover, we can prove the following result:

Lemma 10 (Strong normalization)**.** The reduction $\to$ is strongly normalizing.

Proof. The $[\![\bullet]\!]^{\mathrm{L}}(\bullet)$ is a simulation, and the target system is a call-by-value calculus with a $\Box$ operator, sums, pairs, unit and functions that never returns. It is standard that the system without $\Box$ is strongly normalizing, and the $\lambda^{\neg}_{\Box}$-calculus can be translated into this subsystem by translating $\Box A$ by $A \otimes \mathbb{1}$, $\Box V$ by $(V, *)$ and let $\Box x = t$ in$_{\perp}$ u by let $(x, \_) = t$ in$_{\perp}$ u, and this translation is also a simulation. Thus the reduction $\to$ is strongly normalizing. □

This allows us to prove various property like consistency of the logic, cut-elimination (every term reduce to a term where all cuts are against a variable or a covariable), the subformula property (every sequents admit a proof in which all the formulas that appears in the proof are subformulas of the initial sequent).

The $\lambda^{\neg}_{\Box}$-calculus lacks enough η-rules, thus, we cannot yet prove the computational adequacy of the system, that is, that $\langle \mathrm{true} \mid^+ \gamma\rangle \neq_{\beta\eta} \langle \mathrm{false} \mid^+ \gamma\rangle$. We will get this property later with a model. However, we can already prove, using confluence and strong normalization, that $\langle \mathrm{true} \mid^+ \gamma\rangle \cong \langle \mathrm{false} \mid^+ \gamma\rangle$ is false.

The following lemma will be useful later:

Lemma 11 (Closed normal forms of modal type)**.** If t is a closed term of modal type, then $t \to^* V$ with V some closed value (of the same type).

Proof. Suppose $\cdot \mid \cdot \vdash t : A \mid \cdot$ with $\varpi(A) = \Box$. Let u be the normal form of t: $t \to^* u$. By subject reduction, $\cdot \mid \cdot \vdash u : A \mid \cdot$. Either u is a value and we are done ; or $u = \mu\alpha^+.c$ for some c. In this case, c cannot be written as $\langle v \mid^{\varepsilon} e\rangle$ where v is not a variable and e not a covariable, for then there would be a β-rule which could be applied. For the same reason, either v is a value or e is a stack. Also, v cannot be a variable because there is no variable in the context. This means that $e = \alpha$, v is a value V and $\varepsilon = +$, because α is the only covariable in the context and A is positive. Moreover, A is modal and V is a value: thus, by the lemma 4, α is not free in V. Thus, $u = \mu\alpha^+.\langle V \mid^+ \alpha\rangle \to V$, which is absurd because u is in normal form. □

6. Thunkability

Problems arise when mixing dependent types in presence of effectful terms, in particular involving control operators, as shown by [Her05]. However, we can define a class of terms that are not strictly pure but whose effects are "contained"; for those, the evaluation order does not matter, and the the substitution is well-behaved. Following Fürhmann terminology, we call them thunkable [Füh99]. In particular, all values are thunkable.

Definition 2 (Thunkability)**.** A term t in a context $\Gamma \mid \Theta \vdash \Delta$ is thunkable for a model $[\![\cdot]\!]$ if for every environment e in a context $\Gamma' \mid \Theta' \vdash \Delta'$ and command c in a context $\Gamma'' \mid \Theta'' \vdash \Delta''$, and for all ε_1 and ε_2, one has:

$$[\![\langle t \mid^{\varepsilon_1} \tilde{\mu}a^{\varepsilon_1}.\langle \mu\beta^{\varepsilon_2}.c \mid^{\varepsilon_2} e\rangle\rangle]\!] = [\![\langle \mu\beta^{\varepsilon_2}.\langle t \mid^{\varepsilon_1} \tilde{\mu}a^{\varepsilon_1}.c\rangle \mid^{\varepsilon_2} e\rangle : (\Gamma, \Gamma', \Gamma'' \vdash \Delta, \Delta', \Delta'')]\!]$$

Note that this equality is automatically verified whenever $\varepsilon_1 = -$, $\varepsilon_2 = +$, t is value or e is a stack.

It may be better to give an example: in the more usual call-by-name λ-calculus with let's, if a term t is thunkable then "let $x = t$ in $\lambda y.x$" and "$\lambda y.t$" have the same semantics. It is of course true if all terms are pure: but if we add exceptions to the languages, or non-termination, and if the semantics is expressive enough, then if t is thunkable, it cannot raise an exception, it must terminate and must use the continuations provided by $\mathcal{C}$ linearly.

There exists non thunkable terms in the syntactic model, which interpret terms as their quotient through $=_{\beta\eta}$. For example, consider the term $t = \mu\alpha^+.\langle \text{true} \mid^1 \gamma\rangle$ and the environment $e = \tilde{\mu}b^-.\langle \text{false} \mid^1 \gamma\rangle$, and let c be any command. We have $\cdot \mid \cdot \vdash t : A_+ \vdash \gamma : \mathbb{B}$ and $\cdot \mid \cdot \mid e : B_- \vdash \gamma : \mathbb{B}$ for any A_+ and B_-.

We have

$$\langle t \mid^+ \tilde{\mu}a^+.\langle \mu\beta^-.c \mid^- e\rangle\rangle \to \langle \text{true} \mid^1 \gamma\rangle$$

but

$$\langle \mu\beta^-.\langle t \mid^+ \tilde{\mu}a^+.c\rangle \mid^- e\rangle \to \langle \text{false} \mid^1 \gamma\rangle$$

and these two commands should clearly not be convertible if the calculus is consistent, so t is not thunkable. As remarked earlier, we see that $\mu\alpha^+.\langle \text{true} \mid^+ \gamma\rangle$ is an effectful term because the continuation α is not used linearly.

Thunkability is not a syntactic criterion, thus we cannot rely on it to define a dependent type theory. Moreover, we would like a more type-theoretical approach of thunkability, which is expressive enough for practical use. What we will show is that this theory enables us to build a sound model in which there is a large class of thunkable terms: the terms that are of modal type and typed in a modal context are thunkable, and that this fragment contains intuitionistic logic.

7. Observational equivalence

We devise an observational equivalence for our calculus, in order to build a model. We will define a relation $//_{\gamma:G}$ between commands indexed by the ground context $\cdot \mid \cdot \vdash \gamma : G$ in which

they are defined, where G is a ground type, inspired by [Pit04]. Intuitively, a ground type is a type that one can directly observe and whose values can be compared easily, like booleans and contrary to functions. Formally, it is a type built without using the negative connectives. From now on, G will always denote a ground type. The relation $/\!/$ is defined by $c \,/\!/_{\gamma:G}\, c'$ if there exists a closed value V of type G such that $c \to^* \langle V \mid^+ \gamma\rangle \leftarrow^* c'$. In this case, V is unique, because values of ground types contain no redex. The relation $/\!/$ is saturated, that is, it is stable by antireduction : if $c_1 \,/\!/\, c_2$, $c_1' \to^* c_1$ and $c_2' \to^* c_2$ then $c_1' \,/\!/\, c_2'$.

7.1. Definition of the model

Definition 3 (Orthogonality)**.** Given a type A, for each ground type G we can define an orthogonality relation between pairs of terms of type A and pairs of stacks of type A if A is positive or between pairs of values of type A and pairs of environments of type A if A is negative, where they are all typed in the context $\cdot \mid \cdot \vdash \gamma : G$. We have $(t,t')\perp^+(S,S') \iff \langle t \mid^+ S\rangle \,/\!/_{\gamma:G} \langle t' \mid^+ S'\rangle$ if A is positive and $(V,V')\perp^-(e,e') \iff \langle V \mid^- e\rangle \,/\!/_{\gamma:G} \langle V' \mid^- e'\rangle$ if A is negative.

$/\!/_{\gamma:G}$ induces a Galois connection between relations on terms of type A and relations on stacks of type A in the context $\cdot \mid \cdot \vdash \gamma : G$ if A is positive, and one between relations on values of type A and relations on environments of type A in the same context if A is negative.

For example, if R is a relation on terms of type A in this context, with A positive, $R^{\perp^+}$ is the relation on stacks defined by $SR^{\perp^+}S' \iff \forall (t,t') \in R, (t,t')\perp^+(S,S')$. There is a similarly defined operation $R^{\perp^+}$ if R is a relation on stacks, and we have that $R \subseteq R^{\perp^+\perp^+}$, $R^{\perp^+} = R^{\perp^+\perp^+\perp^+}$, and if $R \subseteq S$ then $S^{\perp^+} \subseteq R^{\perp^+}$. We also have a similar operation $R^{\perp^-}$ if A is negative.

This definition by orthogonality is inspired by classical realizability, which has been introduced by Krivine [Kri09] and adapted by Munch-Maccagnoni [Mun09; Mun17] to a polarised setting.

First, let's define the following relations between values and stacks defined in a context $\cdot \mid \cdot \vdash \gamma : G$: for each type A, if A is positive, then $\|A\|_{\gamma:G}$ is a relation between values, else, if it is negative, it is a relation between stacks of type A. We will at the same time define the relations $\mathbb{T}(A)_{\gamma:G}$, $\mathbb{V}(A)_{\gamma:G}$, $\mathbb{E}(A)_{\gamma:G}$ and $\mathbb{S}(A)_{\gamma:G}$ between terms, values, environments and stacks of type A, and we have that $\|A\|_{\gamma:G} \subseteq \mathbb{V}(A)_{\gamma:G}$ if A is positive and $\|A\|_{\gamma:G} \subseteq \mathbb{S}(A)_{\gamma:G}$ if A is negative.

- $\mathbb{V}(A)_{\gamma:G}$ is the restriction of $\mathbb{T}(A)_{\gamma:G}$ to values;
- $\mathbb{S}(A)_{\gamma:G}$ is the restriction of $\mathbb{E}(A)_{\gamma:G}$ to stacks;
- $\mathbb{T}(A_+)_{\gamma:G} = \mathbb{S}(A_+)^{\perp^+}_{\gamma:G} = \|A_+\|^{\perp^+\perp^+}_{\gamma:G}$;
- $\mathbb{T}(A_-)_{\gamma:G} = \|A_-\|^{\perp^-}_{\gamma:G}$ (note that in this case, $\mathbb{T}(A_-)_{\gamma:G} = \mathbb{V}(A_-)_{\gamma:G}$);
- $\mathbb{E}(A_-)_{\gamma:G} = \mathbb{V}(A_-)^{\perp^-}_{\gamma:G} = \|A_-\|^{\perp^-\perp^-}_{\gamma:G}$;
- $\mathbb{E}(A_+)_{\gamma:G} = \|A_+\|^{\perp^+}_{\gamma:G}$ (and here again, $\mathbb{E}(A_+)_{\gamma:G} = \mathbb{S}(A_+)_{\gamma:G}$);
- $* \,\|\mathbb{1}\|_{\gamma:G}\, *$;
- $(V,W)\, \|A \otimes B\|_{\gamma:G}\, (V',W') \iff V\, \mathbb{V}(A)_{\gamma:G}\, V'$ and $W\, \mathbb{V}(B)_{\gamma:G}\, W'$
- $\iota_i\ V\ \|A_1 \oplus A_2\|\ \iota_i\ V \iff i \in \{1,2\}$ and $V\, \mathbb{V}(A_i)_{\gamma:G}\, V'$

- $\Box V \, \|\Box A\|_{\gamma:G} \, \Box V' \iff V \, \mathbb{V}(A)_{\gamma:G} \, V'$ (note that V and V' are closed, because $\Box V$ and $\Box V'$ are well-typed)
- $[V] \, \|\neg A\|_{\gamma:G} \, [V'] \iff V \, \mathbb{V}(A)_{\gamma:G} \, V'$
- $(S,T) \, \|A \parr B\|_{\gamma:G} \, (S',T') \iff S \, \mathbb{S}(A)_{\gamma:G} \, S'$ and $T \, \mathbb{S}(B)_{\gamma:G} \, T'$

These relations will serve as a base to define a family of equivalence relations between values, terms, stacks, environments and commands. From now, we may drop the subscript $\gamma : G$ when deemed not necessary.

The relation $\mathbb{V}(A)$ and $\mathbb{S}(A)$ can be extended to substitutions targeting a context of the form $\cdot \mid \cdot \vdash \gamma : G$: namely, $\sigma \, [\![\Gamma \mid \Theta \vdash \Delta]\!]^{\gamma:G} \, \sigma'$ if σ and σ' are typed substitutions from the context $\Gamma \mid \Theta \vdash \Delta$ to $\cdot \mid \cdot \vdash \gamma : G$ such that if $x : A \in \Gamma$ then $\sigma(x) \, \mathbb{V}(A) \, \sigma'(x)$, if $x : A \in \Theta$ then $\sigma(x) \, \mathbb{V}(A) \, \sigma'(x)$ and $\sigma(x)$ and $\sigma'(x)$ are typable in the empty context, and if $\alpha : A \in \Delta$ then $\sigma(\alpha) \, \mathbb{S}(A) \, \sigma'(\alpha)$.

Definition 4 (Observational equivalence)**.** It allows to define a relation $\simeq$ by:

- $\Gamma \mid \Theta \vdash V \simeq V' : A; \Delta \iff \forall G, \forall \sigma \, [\![\Gamma \mid \Theta \vdash \Delta]\!]^{\gamma:G} \, \sigma', \sigma(V) \, \mathbb{V}(A)_{\gamma:G} \, \sigma'(V')$
- $\Gamma \mid \Theta \vdash t \simeq t' : A \mid \Delta \iff \forall G, \forall \sigma \, [\![\Gamma \mid \Theta \vdash \Delta]\!]^{\gamma:G} \, \sigma', \sigma(t) \, \mathbb{T}(A)_{\gamma:G} \, \sigma'(t')$
- $\Gamma \mid \Theta; S \simeq S' : A \vdash \Delta \iff \forall G, \forall \sigma \, [\![\Gamma \mid \Theta \vdash \Delta]\!]^{\gamma:G} \, \sigma', \sigma(S) \, \mathbb{S}(A)_{\gamma:G} \, \sigma'(S')$
- $\Gamma \mid \Theta \mid e \simeq e' : A \vdash \Delta \iff \forall G, \forall \sigma \, [\![\Gamma \mid \Theta \vdash \Delta]\!]^{\gamma:G} \, \sigma', \sigma(e) \, \mathbb{E}(A)_{\gamma:G} \, \sigma'(e')$
- $c \simeq c' : (\Gamma \mid \Theta \vdash \Delta) \iff \forall G, \forall \sigma \, [\![\Gamma \mid \Theta \vdash \Delta]\!]^{\gamma:G} \, \sigma', \sigma(c) \, /\!/_{\gamma:G} \, \sigma'(c')$

Remark 12. This definition corresponds to the "logical equivalence" of [Pit04]. He also defines a "contextual equivalence", which is the largest equivalence relation which is adequate, compatible to typing and to substitutions, and a "ciu-equivalence" which has almost the same definition as $\simeq$, but with only one substitution, instead of two that are related. He then procedes to show that they are all equal. We can also define these relations here and show them equal to $\simeq$; the proof will not be given here.

This relation enjoys the following properties:

Definition 5 (Typing compatibility)**.** A relation is compatible with typing if one can apply the typing rules, where terms, values, etc. in the context are replaced by pairs of terms, values, commands, etc. which are $\simeq$.

For example, one then has

$$\frac{}{\cdot \mid \cdot; \alpha \simeq \alpha \vdash \alpha : A}$$

or

$$\frac{c_1 \simeq c_1' : (\Gamma, x : A \mid \Theta \vdash \Delta) \qquad c_2 \simeq c_2' : (\Gamma, y : B \mid \Theta \vdash \Delta)}{\Gamma \mid \Theta; \tilde{\mu}(\iota_1 \; x.c_1 \mid \iota_2 \; y.c_2) \simeq \tilde{\mu}(\iota_1 \; x.c_1' \mid \iota_2 \; y.c_2') : A \oplus B \vdash \Delta}$$

It is easy to show by induction on a typing derivation that relations which are compatible with typing are also reflexive.

For the proof, see section B.1.

Definition 6 (Substitution compatibility)**.** One can extend the relation $\simeq$ to substitutions, saying that $\sigma \simeq \sigma' : (\Gamma \mid \Theta \vdash \Delta) \Rightarrow (\Gamma' \mid \Theta' \vdash \Delta')$ if σ and σ' are typed substitutions of type $(\Gamma \mid \Theta \vdash \Delta) \Rightarrow (\Gamma' \mid \Theta' \vdash \Delta')$ such that

- if $x : A \in \Gamma$ then $\Gamma' \mid \Theta' \vdash \sigma(x) \simeq \sigma'(x) : A; \Delta'$
- if $x : A \in \Gamma$ then $\Gamma'^{\Box} \mid \Theta' \vdash \sigma(x) \simeq \sigma'(x) : A; \cdot$
- if $\alpha : A \in \Gamma$ then $\Gamma' \mid \Theta'; \sigma(\alpha) \simeq \sigma'(\alpha) : A \vdash \Delta'$

Then we say that $\simeq$ is compatible to substitutions if whenever $V \simeq V' : A$, respectively $t \simeq t' : A$, $S \simeq S' : A$, $e \simeq e' : A$ or $c \simeq c'$ in a context $\Gamma \mid \Theta \vdash \Delta$ and $\sigma \simeq \sigma' : (\Gamma \mid \Theta \vdash \Delta) \Rightarrow (\Gamma' \mid \Theta' \vdash \Delta')$, then in the context $\Gamma' \mid \Theta' \vdash \Delta'$ one has $\sigma(V) \simeq \sigma'(V') : A$, $\sigma(t) \simeq \sigma'(t') : A$, $\sigma(S) \simeq \sigma'(S') : A$, $\sigma(e) \simeq \sigma'(e') : A$ and $\sigma(c) \simeq \sigma'(c')$.

For the proof, see section B.2.

Definition 7 (Adequacy)**.** The adequacy property says that $\simeq$ is compatible with $/\!/$. Formally, it says that:

- $\cdot \mid \cdot \vdash V \simeq V' : A; \gamma : G \implies V \,\mathbb{V}\,(A)_{\gamma:G}\, V'$
- $\cdot \mid \cdot \vdash t \simeq t' : A \mid \gamma : G \implies t \,\mathbb{S}\,(A)^{\perp^+}_{\gamma:G}\, t'$
- $\cdot \mid \cdot; S \simeq S' : A \vdash \gamma : G \implies S \,\mathbb{S}\,(A)_{\gamma:G}\, S'$
- $\cdot \mid \cdot \mid e \simeq e' : A \vdash \gamma : G \implies e \,\mathbb{V}\,(A)^{\perp^-}_{\gamma:G}\, e'$
- $c \simeq c' : (\cdot \mid \cdot \vdash \gamma : G) \implies c /\!/_{\gamma:G} c'$

For the proof, see section B.3.

Lemma 13. $\simeq$ is an equivalence relation.

For the proof, see section B.4.

Definition 8 (Compatibility with $=_{\beta\eta}$)**.** If t and u are typed in some context $\Gamma \mid \Theta \vdash \Delta$ and $t =_{\beta\eta} u$ then $\Gamma \mid \Theta \vdash t \simeq u \mid \Delta$. The same goes for values, stacks, environments and commands.

For the proof, see section B.5.

7.2. Consequences

The previous lemmas show that $\simeq$ induces a model for the system $\mathbf{L}^{\Box}_{\mathsf{pol}}$ by quotienting typed terms with $\simeq$. Moreover, this model is sound, as $\langle \text{true} \mid^+ \gamma \rangle \not/\!/ \langle \text{false} \mid^+ \gamma \rangle$, which can be proven through confluence of $\to$. This entails that $=_{\beta\eta}$ is consistent as an equation system.

We know have all what the ingredients needed to prove the first main theorem.

Theorem 1 (Modal terms are thunkable)**.** If $\Gamma^{\Box} \mid \Theta \vdash t : A_{\Box} \mid \cdot$, then t is thunkable in the observational model.

Proof. Let $\Gamma' \mid \Theta' \mid e : B_- \vdash \Delta'$ and $c : (\Gamma'' \mid \Theta'' \vdash \Delta'')$, and let $\sigma \left[\!\!\left[\Gamma^{\Box}, \Gamma', \Gamma'' \mid \Theta, \Theta', \Theta'' \vdash \Delta, \Delta', \Delta''\right]\!\!\right]^{\gamma:G} \sigma'$. Then for all variable x in $\Gamma^{\Box}$ or in Θ, $\sigma(x)$ and $\sigma'(x)$ should by typable in the empty context;

in the second case, by definition, in the first case, because x is of modal types and $\sigma(x)$ and $\sigma'(x)$ are values. Thus $\sigma(t)$ and $\sigma'(t)$ are closed terms of modal type, so by lemma 11 their normal forms are values V and V' respectively.

We know that $\sigma(\langle t \mid^+ \tilde{\mu}a^+.\langle \mu\beta^-.c \mid^- e\rangle\rangle) \to^* \langle \mu\beta^-.\sigma(c)[a := V] \mid^- \sigma(e)\rangle = \bar{\sigma}(\langle \mu\beta^-.c \mid^- e\rangle)$ and $\sigma'(\langle \mu\beta^-.\langle t \mid^+ \tilde{\mu}a^+.c\rangle \mid^- e\rangle) \to^* \langle \mu\beta^-.\sigma'(c)[a := V'] \mid^- \sigma'(e)\rangle = \bar{\sigma}'(\langle \mu\beta^-.c \mid^- e\rangle)$ where $\bar{\sigma}$ and $\bar{\sigma}'$ are the extension in a of σ respectively σ' with V, respectively V'. By saturation, we only have to show them being in relation for $/\!/$.

But $\simeq$ is reflexive, so we only have to prove that $\bar{\sigma}\ [\![\Gamma^{\square}, a : A_{\square}, \Gamma', \Gamma'' \nmid \Theta, \Theta', \Theta'' \vdash \Delta, \Delta', \Delta'']\!]^{\gamma:G}\ \bar{\sigma}'$, that is, $V\ \mathbb{V}(A)\ V'$. We know that $\sigma(t)\ \mathbb{T}(A)\ \sigma'(t')$; let $S\ \mathbb{S}(A)\ S'$. We know that there is W such that $\langle \sigma(t) \mid^+ S\rangle \to^* \langle W \mid^+ \gamma\rangle \leftarrow^* \langle \sigma'(t') \mid^+ S'\rangle$, $\langle \sigma(t) \mid^+ S\rangle \to^* \langle V \mid^+ S\rangle$, $\langle \sigma'(t') \mid^+ S'\rangle \to^* \langle V' \mid^+ S'\rangle$, $\langle W \mid^+ \gamma\rangle$ is a normal form and $\to$ is confluent; thus, $\langle V \mid^+ S\rangle \to^* \langle W \mid^+ \gamma\rangle \leftarrow^* \langle V' \mid^+ S'\rangle$, so $V\ \mathbb{V}(A)\ V'$.

Hence, the result. □

8. Focusing

Before proving the last main theorem, we will need a last technical result. We will need a more general version of strong normalization; namely, that each provable sequent admits a focused proof. Focused proofs correspond to β-short, η-long terms. We will not detail the proof here, because it is an easy adaptation from the one of [Mun17] for this system. The focused system is provided in fig. 3.

Focusing is a proof search technique which has been developed by Andreoli [And92]. It consists on alternating an inversion phase, in which all invertible rules are inverted (a rule is invertible if its premises are derivable from its conclusion) in any order, and a focusing phase, in which one apply the non-invertible rules. The intermediate phase mediate between these two phases.

Lemma 14 (Focusing)**.** For each command $c : (\Gamma \nmid \Theta \vdash \Delta)$, there exists a command c' such that $c =_{\beta\eta} c'$ and $c' : (\Gamma \nmid \Theta \vdash_{inv} \Delta)$. Moreover, inversion steps can be performed in any order.

If V is a value and S is a stack, then they admit similarly a focused derivation.

9. Conservativity

Intuitionistic propositional logic (or at least its fragment without false, but with negation) can be embedded into this calculus by defining a translation $[\![\bullet]\!]$ on formulas such that $[\![\top]\!] = \mathbb{1}$, $[\![A \wedge B]\!] = [\![A]\!] \otimes [\![B]\!]$, $[\![A \vee B]\!] = [\![A \oplus B]\!]$, $[\![A \to B]\!] = \square([\![A]\!] \to [\![B]\!])$ and $[\![\neg A]\!] = \square\neg [\![A]\!]$. It is easy to see that for all formula A, $\varpi([\![A]\!]) = \square$ and that this translation preserves provability. This is why we can think of terms of modal types, typed in a modal context, as intuitionistic. This claim is supported by the fact that, as Gödel already remarked [Göd01], this logic does not prove $A \oplus \square\neg A$ nor $\square\neg\square\neg A \to A$. We will show that this translation also reflects provability.

Theorem 2 (Conservativity)**.** If

- $[\![\Gamma]\!] \nmid \Theta \vdash V : [\![A]\!]\,; \cdot;$

- $[\![\Gamma]\!] \mid \Theta \vdash t : [\![A]\!] \mid \cdot$;
- $[\![\Gamma]\!] \mid \Theta; S : D \vdash \gamma^? : [\![C]\!]$;
- $c : ([\![\Gamma]\!] \mid \Theta \vdash \alpha^? : A)$;

with $\Delta = \gamma^? : [\![C]\!]$ meaning that $\Delta = \cdot$ or $\Delta = \gamma : [\![C]\!]$, D being either of the form $[\![A]\!]$, or $\neg\,[\![A]\!]$, or $\neg\,[\![A]\!] \parr [\![B]\!]$, then V, t, S and c are $=_{\beta\eta}$ to a value, term, stack or command whose proof contains no sequent with two formulas on the right, with the following extra derivation rule.[2]

$$\frac{c : (\Gamma, a : A \mid \Theta \vdash \beta : B)}{\langle \mu(\alpha, \beta).\langle \mu[a].c \mid^- \alpha\rangle \mid^- \gamma\rangle : (\Gamma \mid \Theta \vdash \gamma : \neg A \parr B)}$$

This is indeed a proof that the system $\mathbf{L}^{\Box}_{\mathsf{pol}}$ is conservative over intuitionistic logic, because intuitionistic logic correspond to classical proofs where sequents are restricted to allow only one formula on the right [Gir93].

This translation of intuitionistic logic and this theorem is a variant of the Gödel-McKinsey-Tarski theorem: Gödel remarked that the embedding of intuitionistic logic into the classical S4 logic [Göd01] preserves and reflects provability, and this was later proved by McKinsey and Tarski [MT48]. One can also see the call-by-value translation of intuitionistic logic into linear logic by Girard [Gir87] as a variant of this theorem; indeed, it translates $A \to B$ as $!(A \multimap B)$, and $!$ is a comonad.

For the proof, see section C: it is obtained by focusing.

10. Conclusion

For a summary of my contributions and a discussion on any future work, see the first part of this work.

References

[MT48] J. C. C. McKinsey and Alfred Tarski. "Some Theorems About the Sentential Calculi of Lewis and Heyting". In: *The Journal of Symbolic Logic* 13.1 (1948), pp. 1–15. ISSN: 0022-4812. DOI: 10.2307/2268135.

[Fel+87] Matthias Felleisen et al. "A syntactic theory of sequential control". In: *Theoretical Computer Science* 52.3 (Jan. 1, 1987), pp. 205–237. ISSN: 0304-3975. DOI: 10.1016/0304-3975(87)90109-5.

[Gir87] Jean-Yves Girard. "Linear logic". In: *Theoretical Computer Science* 50.1 (Jan. 1, 1987), pp. 1–101. ISSN: 0304-3975. DOI: 10.1016/0304-3975(87)90045-4.

[Coq89] T. Coquand. *Metamathematical investigations of a calculus of constructions*. report. INRIA, 1989. URL: https://inria.hal.science/inria-00075471 (visited on 07/29/2025).

[2] Otherwise, we cannot introduce the implication; nevertheless, this rule is intuitionistically valid.

[Gri89] Timothy G. Griffin. “A formulae-as-type notion of control”. In: *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. POPL ’90. New York, NY, USA: Association for Computing Machinery, Dec. 1, 1989, pp. 47–58. ISBN: 978-0-89791-343-0. DOI: 10.1145/96709.96714.

[Gir91] Jean-Yves Girard. “A new constructive logic: classic logic”. In: *Mathematical Structures in Computer Science* 1.3 (Nov. 1991), pp. 255–296. ISSN: 0960-1295, 1469-8072. DOI: 10.1017/s0960129500001328.

[And92] Jean-Marc Andreoli. “Logic Programming with Focusing Proofs in Linear Logic”. In: *Journal of Logic and Computation* 2.3 (June 1, 1992), pp. 297–347. ISSN: 0955-792X. DOI: 10.1093/logcom/2.3.297.

[DF92] Oliver Danvy and Andrzex Filinski. “Representing Control: a Study of the CPS Transformation”. In: *Mathematical Structures in Computer Science* 2.4 (Dec. 1992), pp. 361–391. ISSN: 1469-8072, 0960-1295. DOI: 10.1017/S0960129500001535. URL: https://www.cambridge.org/core/journals/mathematical-structures-in-computer-science/article/abs/representing-control-a-study-of-the-cps-transformation/37193FD94F87443338FC7F519783FF0A (visited on 08/08/2025).

[Gir93] Jean-Yves Girard. “On the unity of logic”. In: *Annals of Pure and Applied Logic* 59.3 (Feb. 16, 1993), pp. 201–217. ISSN: 0168-0072. DOI: 10.1016/0168-0072(93)90093-S.

[OR94] Vincent van Oostrom and Femke van Raamsdonk. “Weak orthogonality implies confluence: The higher-order case”. In: *Logical Foundations of Computer Science*. Ed. by Anil Nerode and Yu. V. Matiyasevich. Berlin, Heidelberg: Springer, 1994, pp. 379–392. ISBN: 978-3-540-48442-4. DOI: 10.1007/3-540-58140-5_35.

[DJS97] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. “A New Deconstructive Logic: Linear Logic”. In: *The Journal of Symbolic Logic* 62.3 (1997), pp. 755–807. ISSN: 0022-4812. DOI: 10.2307/2275572.

[Füh99] Carsten Führmann. “Direct Models of the Computational Lambda-calculus”. In: *Electronic Notes in Theoretical Computer Science*. MFPS XV, Mathematical Foundations of Progamming Semantics, Fifteenth Conference 20 (Jan. 1, 1999), pp. 245–292. ISSN: 1571-0661. DOI: 10.1016/S1571-0661(04)80078-1.

[CH00] Pierre-Louis Curien and Hugo Herbelin. “The duality of computation”. In: *SIGPLAN Not.* 35.9 (Sept. 1, 2000), pp. 233–243. ISSN: 0362-1340. DOI: 10.1145/357766.351262.

[Göd01] Kurt Gödel. “An interpretation of the intuitionistic prepositional calculus (1933f)”. In: *Kurt Gödel: Collected Works: Volume I: Publications 1929-1936*. Ed. by Kurt Gödel et al. Oxford, New York: Oxford University Press, July 5, 2001. ISBN: 978-0-19-514720-9.

[Pit04] Andrew Pitts. “Typed Operational Reasoning”. In: Benjamin C. Pierce. *Advanced Topics in Types and Programming Languages*. The MIT Press, Nov. 2004. ISBN: 978-0-262-16228-9. URL: https://www.cis.upenn.edu/~bcpierce/attapl/.

[Her05] Hugo Herbelin. "On the Degeneracy of Σ-Types in Presence of Computational Classical Logic". In: *Typed Lambda Calculi and Applications*. 7th International Conference, TLCA 2005. Vol. 3461. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 209–220. ISBN: 978-3-540-25593-2. DOI: 10.1007/11417170_16.

[App07] Andrew Appel. *Compiling with Continuations*. Cambridge: Cambridge University Press, 2007. ISBN: 978-0-521-03311-4. URL: https://www.cambridge.org/universitypress/subjects/computer-science/programming-languages-and-applied-logic/compiling-continuations (visited on 07/31/2025).

[Kri09] Jean-Louis Krivine. "Realizability in classical logic". In: *Panoramas et synthèses* 27 (2009), pp. 197–229. URL: https://hal.science/hal-00154500.

[Mun09] Guillaume Munch-Maccagnoni. "Focalisation and Classical Realisability (version with appendices)". In: 18th EACSL Annual Conference on Computer Science Logic - CSL 09. Vol. 5771. Springer-Verlag, 2009, p. 409. DOI: 10.1007/978-3-642-04027-6_30.

[CFM16] Pierre-Louis Curien, Marcelo Fiore, and Guillaume Munch-Maccagnoni. "A theory of effects and resources: adjunction models and polarised calculi". In: *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '16: The 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. St. Petersburg FL USA: ACM, Jan. 11, 2016, pp. 44–56. ISBN: 978-1-4503-3549-2. DOI: 10.1145/2837614.2837652. URL: https://dl.acm.org/doi/10.1145/2837614.2837652 (visited on 07/04/2025).

[Kav17] G. A. Kavvos. "Dual-context calculi for modal logic". In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). Reykjavik, Iceland: IEEE, June 2017, pp. 1–12. ISBN: 978-1-5090-3018-7. DOI: 10.1109/LICS.2017.8005089.

[Mun17] Guillaume Munch-Maccagnoni. *Note on Curry's style for Linear Call-by-Push-Value*. 2017. URL: https://inria.hal.science/hal-01528857v3.

[Bow18] William J. Bowman. "Compiling with dependent types". PhD thesis. Northeastern University, 2018. DOI: 10.17760/D20316239.

[Con+19] Youyou Cong et al. "Compiling with continuations, or without? whatever." In: *Proceedings of the ACM on Programming Languages* 3 (ICFP July 26, 2019), pp. 1–28. ISSN: 2475-1421. DOI: 10.1145/3341643.

[Miq19] Étienne Miquey. "A Classical Sequent Calculus with Dependent Types". In: *ACM Trans. Program. Lang. Syst.* 41.2 (Mar. 15, 2019), 8:1–8:47. ISSN: 0164-0925. DOI: 10.1145/3230625.

A. Appendix: Proofs for the CPS transformation

A.1. Proof of lemma 1

This proof is inspired by the proof of [Con+19].

First, we need a lemma saying that $[\![\bullet]\!]^\lambda$ is compatible with the substitution, that is, $[\![t]\!]^\lambda_\perp\,[x := [\![V]\!]^\lambda_\mathsf{v}] = [\![t[x := V]]\!]^\lambda_\perp$, and so on. It is obvious.

For reductions like for functions types, we have, for example,

$$\begin{aligned}
&[\![(\lambda x.t)\ V]\!]^\lambda_\mathsf{t}\,(M)\\
&\qquad = (\lambda^\perp k.k^\perp\ ([\![V]\!]^\lambda_\mathsf{v}, \lambda^\perp y.M^\perp\ y))^\perp\ (\lambda^\perp(x,k).\,[\![t]\!]^\lambda_\mathsf{t}\,(k))\\
&\qquad \to^* [\![t]\!]^\lambda_\mathsf{t}\,(\lambda^\perp y.M^\perp\ y)[x := [\![V]\!]^\lambda_\mathsf{v}]\\
&\qquad = [\![t[x := V]]\!]^\lambda_\mathsf{t}\,(\lambda^\perp y.M^\perp\ y)\\
&\qquad \to^* [\![t[x := V]]\!]^\lambda_\mathsf{t}\,(M)
\end{aligned}$$

It also works the same for other reductions, except for $\mathcal{C}$. For $\mathcal{C}$, on the one hand, we have that $[\![\mathcal{C}\ (\lambda^\perp k.k^\perp\ V)]\!]^\lambda_\mathsf{t}\,(M) = (\lambda^\perp v.v^\perp\ M)^\perp\ (\lambda^\perp k.k^\perp\ \ [\![V]\!]^\lambda_\mathsf{v}) \to^* M^\perp\ \ [\![V]\!]^\lambda_\mathsf{v} = [\![V]\!]^\lambda_\mathsf{t}\,(M)$.

On the second hand, let us define a translation $[\![F[\_]]\!]^\lambda_\mathsf{F}$ on elementary contexts $F[\_]$ that verifies $[\![F[t]]\!]^\lambda_\mathsf{t}\,(M) = [\![t]\!]^\lambda_\mathsf{t}\,([\![F[\_]]\!]^\lambda_\mathsf{F}\,(M))$, inspired by [Con+19].

- $[\![[\_]\ V]\!]^\lambda_\mathsf{F}\,(M) = \lambda^\perp k.k^\perp\ ([\![V]\!]^\lambda_\mathsf{v}, \lambda^\perp x.M^\perp\ x)$
- $[\![\text{let } x = [\_] \text{ in } t]\!]^\lambda_\mathsf{F}\,(M) = \lambda^\perp x.\,[\![t]\!]^\lambda_\mathsf{t}\,(M)$
- $[\![\text{let } * = [\_] \text{ in } t]\!]^\lambda_\mathsf{F}\,(M) = \lambda^\perp z.\,\text{let } * = [\![t]\!]^\lambda_\mathsf{t}\,(M) \text{ in}_\perp$
- $[\![\text{let } (x,y) = [\_] \text{ in } t]\!]^\lambda_\mathsf{F}\,(M) = \lambda^\perp (x,y).\,[\![t]\!]^\lambda_\mathsf{t}\,(M)$
- $[\![\text{match } [\_] \text{ with } \{\iota_1\ x.t \mid \iota_2\ y.u\}]\!]^\lambda_\mathsf{F}\,(M) =$
  $\lambda^\perp z.\,\text{match } z \text{ with}_\perp \{\iota_1\ x.\,[\![t]\!]^\lambda_\mathsf{t}\,(M) \mid \iota_2\ y.\,[\![u]\!]^\lambda_\mathsf{t}\,(M)\}$
- $[\![\mathcal{C}\ [\_]]\!]^\lambda_\mathsf{F}\,(M) = \lambda^\perp v.v^\perp\ M$

Then we have:

$$\begin{aligned}
&\left[\!\!\left[F[\mathcal{C}\ (\lambda^\perp k.t)]\right]\!\!\right]^\lambda_\mathsf{t}\,(M)\\
&\qquad = \left[\!\!\left[\mathcal{C}\ (\lambda^\perp k.t)\right]\!\!\right]^\lambda_\mathsf{t}\,([\![F[_]]\!]^\lambda_\mathsf{F}\,(M))\\
&\qquad = (\lambda^\perp v.v^\perp\ ([\![F[_]]\!]^\lambda_\mathsf{F}\,(M)))^\perp\ (\lambda^\perp k.\,[\![t]\!]^\lambda_\perp)\\
&\qquad \to^* [\![t]\!]^\lambda_\perp\,[k := [\![[_]]\!]^\lambda_\mathsf{F}\,(M)]
\end{aligned}$$

and on the other side

$$\begin{aligned}
&\left[\!\!\left[\mathcal{C}\ (\lambda^\perp j.t[k := \lambda^\perp x.\,\text{let } y = F[x] \text{ in}_\perp\ j^\perp\ y])\right]\!\!\right]^\lambda_\mathsf{t}\,(M)\\
&\qquad = (\lambda^\perp v.v^\perp\ M)^\perp\ (\lambda^\perp j.\,[\![t]\!]^\lambda_\perp\,[k := \lambda^\perp x.\,[\![F[x]]\!]^\lambda_\mathsf{t}\,(\lambda^\perp y.j^\perp\ y)])\\
&\qquad \to^* [\![t]\!]^\lambda_\perp\,[k := \lambda^\perp x.\,[\![F[x]]\!]^\lambda_\mathsf{t}\,(\lambda^\perp y.M^\perp\ y)]\\
&\qquad = [\![t]\!]^\lambda_\perp\,[k := \lambda^\perp x.\,[\![F[_]]\!]^\lambda_\mathsf{F}\,(\lambda^\perp y.M^\perp\ y)^\perp\ x]\\
&\qquad \to^* [\![t]\!]^\lambda_\perp\,[k := [\![F[_]]\!]^\lambda_\mathsf{F}\,(M)]
\end{aligned}$$

Thus, both terms are in relation by $\cong$.

A.2. Proof of lemma 5

We prove this statement by induction on the derivation.

Three cases are not straightforward: the first one is the renaming rule. We must show that $\sigma \circ \theta : (\Gamma \mid \Theta \vdash \Delta) \Rightarrow (\Gamma'' \mid \Theta'' \vdash \Delta'')$. It is indeed the case: if $x : A \in \Gamma$, $\theta(x) : A \in \Gamma'$ then $\Gamma'' \mid \Theta'' \vdash \sigma(\theta(x)) : A; \Delta''$, and the same goes for $\alpha : A \in \Delta$. If $x : A \in \Theta$, $\theta(x) : A \in \Theta'$ thus $\Gamma''^{\Box} \mid \Theta'' \vdash \sigma(\theta(x)) : A \vdash \cdot$. Hence, $\sigma \circ \theta$ is indeed a substitution of the good type.

The second case is the left rule for $\Box$. If $\sigma : (\Gamma^{\Box} \mid \Theta \vdash \cdot) \Rightarrow (\Gamma' \mid \Theta' \vdash \Delta')$, we can restrict σ to be a substitution $\sigma^{\Box} : (\Gamma^{\Box} \mid \Theta \vdash \cdot) \Rightarrow (\Gamma'^{\Box} \mid \Theta' \vdash \cdot)$: indeed, if $x : \Box A \in \Gamma^{\Box}$ then $\Gamma' \mid \Theta' \vdash \sigma(x) : \Box A; \Delta'$ but by lemma 4, we have $\Gamma'^{\Box} \mid \Theta' \vdash \sigma(x) : \Box A; \cdot$, and if $x : A \in \Theta$ then $\Gamma'^{\Box} \mid \Theta' \vdash \sigma(x) : A; \cdot$ because σ is a substitution. Therefore, if $\Gamma^{\Box} \mid \Theta \vdash V : A; \cdot$ we have by induction that $\Gamma'^{\Box} \mid \Theta' \vdash \sigma^{\Box}(V) : A; \cdot$, hence $\Gamma'^{\Box} \mid \Theta' \vdash \Box\sigma^{\Box}(V) : \Box A; \cdot$ and $\Gamma' \mid \Theta' \vdash \Box\sigma^{\Box}(V) : \Box A; \Delta$ by weakening. But $\sigma(\Box V) = \Box\sigma^{\Box}(V)$, therefore $\Gamma' \mid \Theta' \vdash \sigma(\Box V) : \Box A; \Delta'$.

The last case is the one of rules with several premises. We will deal with the case of the left rule for $\otimes$. Suppose $\sigma : (\Gamma_1, \Gamma_2 \mid \Theta_1, \Theta_2 \vdash (V, W) : A \otimes B; \Delta_1, \Delta_2) \Rightarrow (\Gamma' \mid \Theta' \vdash \Delta')$. We have $\Gamma' \mid \Theta' \vdash \sigma(V) : A; \Delta'$ and $\Gamma' \mid \Theta' \vdash \sigma(V) : A; \Delta'$. We cannot apply the introduction rule for $\otimes$ yet, because the contexts are not disjoint. Therefore, we will first apply a renaming on both side : the θ_i defined by $\theta_i(x) = x_i$ and $\theta_i(\alpha) = \alpha_i$ are renaming into the context $(\Gamma'_i \mid \Theta'_i \vdash \Delta'_i)$, with Γ'_i the context Γ' where variables have been renamed by indexing them with i, and so on for Θ'_i and Δ'_i. We thus have $\Gamma'_1, \Gamma'_2 \mid \Theta'_1, \Theta'_2 \vdash (\theta_1 \circ \sigma(V), \theta_2 \circ \sigma(W)) : A \otimes B; \Delta'_1, \Delta'_2$.

Lastly, we apply the renaming θ which goes to the context $\Gamma' \mid \Theta' \vdash \Delta'$, defined by $\theta(x_i) = x$ and $\theta(\alpha_i) = \alpha$. We hence have $\Gamma' \mid \cdot \vdash \theta((\theta_1 \circ \sigma(V), \theta_2 \circ \sigma(W))) : A \otimes B; \Delta'$. But $\theta \circ \theta_i$ is the identity, so finally we get $\Gamma' \mid \Theta' \vdash \sigma((V, W)) : A \otimes B; \Delta'$.

A.3. Proof of lemma 8

The proof will go on as in section A.1.

We first need an obvious substitution lemma, similar to the one in the aforementioned proof.

Then, for the main proof, we have, for example, for functions:

$$\begin{aligned}
&(\!|(\lambda a.t)\ V|\!)_{\mathsf{t}}(S) \\
&\qquad = \langle \{\mu(\alpha,\beta).\langle \mu[a].(\!|t|\!)_{\mathsf{t}}(\beta) \mid^{-} \alpha\rangle\} \mid^{+} \tilde{\mu}\{f\}.\langle f \mid^{-} ([(\!|V|\!)_{\mathsf{v}}], S)\rangle\rangle \\
&\qquad\qquad \to^{*} (\!|t|\!)_{\mathsf{t}}(S)[x := (\!|V|\!)_{\mathsf{v}}] \\
&\qquad\qquad = (\!|t[x := V]|\!)_{\mathsf{t}}(S)
\end{aligned}$$

The other β-reductions, except for $\mathcal{C}$, are similar.

For $\mathcal{C}$, first, notice that

$$\begin{aligned}
&(\!|\mathcal{C}\ (\lambda k^{\perp}.k^{\perp}\ V)|\!)_{\mathsf{t}}(S) \\
&\qquad = \langle \{\mu[k].\langle k \mid^{+} \tilde{\mu}\{f\}.\langle f \mid^{-} [(\!|V|\!)_{\mathsf{v}}]\rangle\rangle\} \mid^{+} \tilde{\mu}\{x\}.\langle x \mid^{-} [\{\mu[y].\langle y \mid^{+} S\rangle\}]\rangle\rangle \\
&\qquad\qquad \to^{*} \langle (\!|V|\!)_{\mathsf{v}} \mid^{+} S\rangle \\
&\qquad\qquad = (\!|V|\!)_{\mathsf{t}}(S)
\end{aligned}$$

Then, we will again define $(\!|F[\_]|\!)_{\mathsf{F}}(S)$ for an elementary context $F[\_]$ and a stack S. We want to ensure the property that for all t, $(\!|F[t]|\!)_{\mathsf{t}}(S) = (\!|t|\!)_{\mathsf{t}}((\!|F[\_]|\!)_{\mathsf{F}}(S))$:

$$(\!|[\_]\ V|\!)_{\mathsf{F}}(S) = \tilde{\mu}\{f\}.\langle f \mid^{-} ([(\!|V|\!)_{\mathsf{v}}], S)\rangle \qquad (\!|\text{let } x = [\_] \text{ in } t|\!)_{\mathsf{F}}(S) = \tilde{\mu}x.(\!|t|\!)_{\mathsf{t}}(S)$$

$$(\!|\text{let } (x,y) = [\_] \text{ in } t|\!)_{\mathsf{F}}(S) = \tilde{\mu}(x,y).(\!|t|\!)_{\mathsf{t}}(S)$$

$$(\!|\text{match } [\_] \text{ with } \{\iota_1\ x.t \mid \iota_2\ y.u\}|\!)_{\mathsf{F}}(S) = \tilde{\mu}(\iota_1\ x.(\!|t|\!)_{\mathsf{t}}(S) \mid \iota_2\ y.(\!|u|\!)_{\mathsf{t}}(S))$$

$$(\!|\text{let } \Box x = [\_] \text{ in } t|\!)_{\mathsf{F}}(S) = \tilde{\mu}\Box x.(\!|t|\!)_{\mathsf{t}}(S) \qquad (\!|\mathcal{C}\ [\_]|\!)_{\mathsf{F}}(S) = \tilde{\mu}\{x\}.\langle x \mid^{-} [\{\mu[y].\langle y \mid^{+} S\rangle\}]\rangle$$

Then we have

$$\begin{aligned}
&(\!|F[\mathcal{C}\ (\lambda^{\perp}k.t)]|\!)_{\mathsf{F}}(S) \\
&\qquad = (\!|\mathcal{C}\ (\lambda^{\perp}k.t)|\!)_{\mathsf{t}}((\!|F[_]|\!)_{\mathsf{F}}(S)) \\
&\qquad = \langle\{\mu[k].(\!|t|\!)_{\perp}\} \mid^{+} \tilde{\mu}\{x\}.\langle x \mid^{-} [\{\mu[y].\langle y \mid^{+} (\!|F[_]|\!)_{\mathsf{F}}(S)\rangle\}]\rangle\rangle \\
&\qquad \to (\!|t|\!)_{\perp}[k := \{\mu[y].\langle y \mid^{+} (\!|F[_]|\!)_{\mathsf{F}}(S)\rangle\}]
\end{aligned}$$

and

$$\begin{aligned}
&(\!|\mathcal{C}\ (\lambda^{\perp}j.t[k := \lambda^{\perp}x.\,\text{let } y = F[x] \text{ in}_{\perp}\ j^{\perp}\ y])|\!)_{\mathsf{t}}(S) \\
&= \langle\{\mu[j].(\!|t|\!)_{\perp}[k := \{\mu[x].(\!|F[x]|\!)_{\mathsf{t}}(\tilde{\mu}y^{+}.\langle j \mid^{+} \tilde{\mu}\{i\}.\langle i \mid^{-} [y]\rangle\rangle)\}]\} \mid^{+} \tilde{\mu}\{x\}.\langle x \mid^{-} [\{\mu[y].\langle y \mid^{+} S\rangle\}]\rangle\rangle \\
&= \langle\{\mu[j].(\!|t|\!)_{\perp}[k := \{\mu[x].\langle x \mid^{+} (\!|F[_]|\!)_{\mathsf{F}}(\tilde{\mu}y^{+}.\langle j \mid^{+} \tilde{\mu}\{i\}.\langle i \mid^{-} [y]\rangle\rangle)\}]\} \mid^{+} \tilde{\mu}\{x\}.\langle x \mid^{-} [\{\mu[y].\langle y \mid^{+} S\rangle\}]\rangle\rangle \\
&\qquad \to^{*} (\!|t|\!)_{\perp}[k := \{\mu[x].\langle x \mid^{+} (\!|F[_]|\!)_{\mathsf{F}}(\tilde{\mu}y^{+}.\langle y \mid^{+} S\rangle)\rangle\}] \\
&\qquad \to^{*} (\!|t|\!)_{\perp}[k := \{\mu[x].\langle x \mid^{+} (\!|F[_]|\!)_{\mathsf{F}}(S)\rangle\}]
\end{aligned}$$

Thus, both are in relation by $\cong$.

A.4. Proof of lemma 7

For types, it is straightforward.

We will not prove all the cases for redexes, but only the translation of $\mathcal{C}\ t$ and functions.

$$\begin{aligned}
&[\![(\!|\mathcal{C}\ t|\!)_{\mathsf{t}}(S)]\!]^{\mathrm{L}}_{\mathsf{c}} \\
&\qquad = [\![(\!|t|\!)_{\mathsf{t}}(\tilde{\mu}\{x\}.\langle x \mid^{-} [\{\mu[y].\langle y \mid^{+} S\rangle\}]\rangle)]\!]^{\mathrm{L}}_{\mathsf{c}} \\
&\qquad \to^{*} [\![t]\!]^{\lambda}_{\mathsf{t}}\,([\![\tilde{\mu}\{x\}.\langle x \mid^{-} [\{\mu[y].\langle y \mid^{+} S\rangle\}]\rangle]\!]^{\mathrm{L}}_{\mathsf{s}}) \\
&\qquad = [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda^{\perp}x.x^{\perp}\ (\lambda^{\perp}y.([\![S]\!]^{\mathrm{L}}_{\mathsf{s}})^{\perp}\ y)) \\
&\qquad \to [\![t]\!]^{\lambda}_{\mathsf{t}}\,(\lambda^{\perp}x.x^{\perp}\ ([\![S]\!]^{\mathrm{L}}_{\mathsf{s}})^{\perp}) \\
&\qquad = [\![\mathcal{C}\ t]\!]^{\lambda}_{\mathsf{t}}\,([\![S]\!]^{\mathrm{L}}_{\mathsf{s}})
\end{aligned}$$

$[\![(\!|\lambda a.t|\!)_{\mathsf{v}}]\!]^{\mathrm{L}}_{\mathsf{v}}$

$$
\begin{aligned}
&= [\![\{\mu(\alpha,\beta).\langle \mu[a].(\!|t|\!)_{\mathsf{t}}(\beta) \mid^{-} \alpha\rangle\}]\!]^{\mathrm{L}}_{\mathsf{v}} \\
&= \lambda^{\perp}(\alpha,\beta).(\lambda^{\perp} a.\, [\![(\!|t|\!)_{\mathsf{t}}(\beta)]\!]^{\mathrm{L}}_{\mathsf{c}})^{\perp}\ \alpha \\
&\to^{*} \lambda^{\perp}(\alpha,\beta).\, [\![t]\!]^{\lambda}_{\mathsf{t}}\, ([\![\beta]\!]^{\mathrm{L}}_{\mathsf{s}})[a := \alpha] \\
&= \lambda^{\perp}(a,\beta).\, [\![t]\!]^{\lambda}_{\mathsf{t}}\, (\beta) \\
&= [\![\lambda a.t]\!]^{\lambda}_{\mathsf{v}}
\end{aligned}
$$

$[\![(\!|t\ V|\!)_{\mathsf{t}}(S)]\!]^{\mathrm{L}}_{\mathsf{c}}$

$$
\begin{aligned}
&= [\![(\!|t|\!)_{\mathsf{t}}(\tilde{\mu}\{f\}.\langle f \mid^{-} ((\!|V|\!)_{\mathsf{v}}, S)\rangle)]\!]^{\mathrm{L}}_{\mathsf{c}} \\
&\to^{*} [\![t]\!]^{\lambda}_{\mathsf{t}}\, (\lambda^{\perp} f.f^{\perp}\ ([\![(\!|V|\!)_{\mathsf{v}}]\!]^{\mathrm{L}}_{\mathsf{v}}, [\![S]\!]^{\mathrm{L}}_{\mathsf{s}})) \\
&\to^{*} [\![t]\!]^{\lambda}_{\mathsf{t}}\, (\lambda^{\perp} f.f^{\perp}\ ([\![V]\!]^{\lambda}_{\mathsf{v}}, [\![S]\!]^{\mathrm{L}}_{\mathsf{s}})) \\
&= [\![t\ V]\!]^{\lambda}_{\mathsf{t}}\, ([\![S]\!]^{\mathrm{L}}_{\mathsf{s}})
\end{aligned}
$$

B. Appendix: Proofs for the observational equivalence

B.1. Typing compatibility proof (definition 5)

Proof. We will show some key cases.

$\Box x$: We want to show that $\cdot \nmid x : A \vdash x \simeq x : A$. Given $\sigma\ [\![\cdot \nmid x : A \vdash \cdot]\!]\ \sigma'$, we have $\sigma(x)\ \mathbb{V}\,(A)\ \sigma'(x)$ by definition.

$\Box V$: Suppose that $\Gamma^{\Box} \nmid \Theta \vdash V \simeq V' : A; \cdot$. Then for all $\sigma\ [\![\Gamma^{\Box} \nmid \Theta \vdash \cdot]\!]\ \sigma'$, by definition we have $\sigma(V)\ \mathbb{V}\,(A)\ \sigma'(V')$ thus by definition of $\|\Box A\|$ we have $\sigma(\Box V) = \Box\sigma(V)\ \|\Box A\|\ \Box\sigma'(V') = \sigma'(\Box V')$, and $\|\Box A\| \subseteq \mathbb{V}\,(\Box A)$. Thus, $\Gamma^{\Box} \nmid \Theta \vdash \Box V \simeq \Box V' : \Box A; \cdot$.

$\tilde{\mu}\Box x.c$: Suppose that $c \simeq c' : (\Gamma \nmid \Theta, x : A \vdash \Delta)$. We want to show that $\Gamma \nmid \Theta; \tilde{\mu}\Box x.c \simeq \tilde{\mu}\Box x.c' : \Box A \vdash \Delta$. Let $\sigma\ [\![\Gamma \nmid \Theta \vdash \Delta]\!]\ \sigma'$. $\mathbb{S}\,(\Box A) = \|\Box A\|^{\perp^{+}}$, so we must show is that if $V\ \|\Box A\|\ V'$ then $\langle V \mid \sigma(\tilde{\mu}\Box x.c)\rangle\ //\langle V' \mid \sigma'(\tilde{\mu}\Box x.c')\rangle$.

We know that V and V' can be written as $\Box W$ and $\Box W'$ with $W\ \mathbb{V}\,(A)\ W'$. Thus, we can extend σ and σ' by defining $\bar{\sigma}(x) = W$ and $\bar{\sigma}'(x) = W'$, and we have $\bar{\sigma}\ [\![\Gamma \nmid \Theta, x : A \vdash \Delta]\!]\ \bar{\sigma}'$. Thus, $\bar{\sigma}(c)\ //\ \bar{\sigma}'(c')$ by the induction hypothesis. That means that $\sigma(c)[x := W]\ //\ \sigma'(c')[x := W]$. But $\langle V \mid \sigma(\tilde{\mu}\Box x.c)\rangle \to \sigma(c)[x := W]$ and $\langle V' \mid \sigma'(\tilde{\mu}\Box x.c')\rangle \to \sigma'(c')[x := W']$, so by saturation, we finally get that $\langle V \mid \sigma(\tilde{\mu}\Box x.c)\rangle\ //\langle V' \mid \sigma'(\tilde{\mu}\Box x.c')\rangle$.

S: If $\Gamma \nmid \Theta; S \simeq S' : A_{-} \vdash \Delta$, we want to show that $\Gamma \nmid \Theta \mid S \simeq S' : A_{-} \vdash \Delta$ as environments. A is negative, so $\mathbb{S}\,(A)$ is the restriction of $\mathbb{E}\,(A)$ to stacks, so it is true.

$\mu\alpha^{\varepsilon}.c$: if $c \simeq c' : (\Gamma \nmid \Theta \vdash \Delta, \alpha : A_{\varepsilon})$, we want to show that $\Gamma \nmid \Theta \vdash \mu\alpha^{+}.c \simeq \mu\alpha^{+}.c' : A_{+} \mid \Delta$ if A is positive and $\Gamma \nmid \Theta \vdash \mu\alpha^{-}.c \simeq \mu\alpha^{-}.c' : A_{-}; \Delta$ if A is negative. Let $\sigma\ [\![\Gamma \nmid \Theta \vdash \Delta]\!]^{\gamma:G}\ \sigma'$. Whether A is positive or negative, we want to show that $\sigma(\mu\alpha^{\varepsilon}.c)\mathbb{S}\,(A)^{\perp^{\varepsilon}}\ \sigma'(\mu\alpha^{\varepsilon}.c')$. Thus, let S and S' such that $S\mathbb{S}\,(A)\,S'$. We want to show that $\langle \mu\alpha^{\varepsilon}.\sigma(c) \mid^{\varepsilon} S\rangle\ //\langle \mu\alpha^{\varepsilon}.\sigma'(c') \mid^{\varepsilon} S'\rangle$.

We can extend σ, respectively σ' in α by S or S', and the resulting substitutions verify $\bar{\sigma}\,[\![\Gamma \mid \Theta \vdash \Delta, \alpha : A]\!]\,\bar{\sigma}'$. We know that by saturation, it is enough to show that $\sigma(c)[\alpha := S] \mathbin{/\!/} \sigma'(c')[\alpha := S']$, but this means $\bar{\sigma}(c) \mathbin{/\!/} \bar{\sigma}'(c')$. It is true because $c \simeq c'$.

$\langle t \mid^+ S\rangle$: If $\Gamma \mid \Theta \vdash t \simeq: A \mid \Delta$ and $\Gamma \mid \Theta; S \simeq S' \vdash \Delta$ where A is positive, then for all $\sigma\,[\![\Gamma \mid \Theta \vdash \Delta]\!]\,\sigma'$, $\sigma(t)\,\mathbb{T}\,(A)\,\sigma'(t')$ and $\sigma(S)\,\mathbb{S}\,(A)\,\sigma'(S')$, but $\mathbb{T}\,(A) = \mathbb{S}\,(A)^{\perp^+}$ so by definition of $\perp^+$, $\langle t \mid^+ S\rangle \mathbin{/\!/} \langle t' \mid^+ S'\rangle$.

$\theta(c)$: Suppose that $c \simeq c' : (\Gamma \mid \Theta \vdash \Delta)$ and $\theta \in \mathfrak{R}(\Gamma \mid \Theta \vdash \Delta \Rightarrow \Gamma' \mid \Theta' \vdash \Delta')$. We want to show that $\theta(c) \simeq \theta(c') : (\Gamma' \mid \Theta' \vdash \Delta')$. Let $\sigma\,[\![\Gamma' \mid \Theta' \vdash \Delta]\!]\,\sigma'$. Then $\sigma \circ \theta\,[\![\Gamma \mid \Theta \vdash \Delta]\!]\,\sigma' \circ \theta$. Thus, we can apply them on $c \simeq c'$ to get that $\sigma(\theta(c)) \mathbin{/\!/} \sigma'(\theta(c'))$.

□

B.2. Substitution compatibility proof (definition 6)

Proof. Suppose $c \simeq c' : (\Gamma \mid \Theta \vdash \Delta)$ and $\sigma \simeq \sigma' : (\Gamma' \mid \Theta \vdash \Delta) \Rightarrow (\Gamma' \mid \Theta' \vdash \Delta')$. Let's show that $\sigma(c) \simeq \sigma'(c') : (\Gamma' \mid \Theta' \vdash \Delta')$, that is, for all $\psi\,[\![\Gamma' \mid \Theta' \vdash \Delta']\!]\,\psi'$, $\psi \circ \sigma(c) \mathbin{/\!/} \psi' \circ \sigma'(c')$. We want to apply the definition of $c \simeq c'$ with $\psi \circ \sigma$ and $\psi' \circ \sigma'$, so we have to prove that $\psi \circ \sigma\,[\![\Gamma \mid \Theta \vdash \Delta]\!]\,\psi' \circ \sigma'$.

$\psi \circ \sigma$ and $\psi' \circ \sigma'$ are substitutions with the good type ; now let $\alpha : A \in \Delta$. Then by definition of $\sigma \simeq \sigma'$, $\Gamma' \mid \Theta'; \sigma(\alpha) \simeq \sigma'(\alpha) \vdash \Delta'$. Thus, by definition of $\psi\,[\![\Gamma' \mid \Theta' \vdash \Delta']\!]\,\psi'$, $\psi \circ \sigma(\alpha)\,\mathbb{S}\,(A)\,\psi' \circ \sigma'(\alpha)$. The proof works for $x : A \in \Gamma$ or $x : A \in \Theta$ (but we have to restrict ψ here), so indeed $\psi \circ \sigma\,[\![\Gamma \mid \Theta \vdash \Delta]\!]\,\psi' \circ \sigma'$. □

B.3. Adequacy proof (definition 7)

Proof. We will show the proof for the case of commands ; the other cases are proved similarly. Suppose that $c \simeq c' : (\cdot \mid \cdot \vdash \gamma : G)$.

Let $\sigma(\gamma) = \sigma'(\gamma) = \gamma$. Then $\sigma\,[\![\cdot \mid \cdot \vdash \gamma : G]\!]^{\gamma:G}\,\sigma'$. Indeed, let's show that $\gamma\,\mathbb{S}\,(G)\,\gamma$, that is, because G is positive, for every $V\,\|G\|\,V'$ we have $\langle V \mid^+ \gamma\rangle \mathbin{/\!/} \langle V' \mid^+ \gamma\rangle$.

We will prove by induction on G the following three properties:

H1 If $V\,\|G\|_{\gamma:G}\,V'$ then $V = V'$.

H2 $\gamma\,\mathbb{S}\,(G)_{\gamma:G}\,\gamma$.

H3 If $V\,\mathbb{V}\,(G)_{\gamma:G}\,V'$ then $V = V'$.

First, we will show that if H1 is true for G, then H2 and H3 too. Indeed, suppose $V\,\|G\|\,V'$. Then $V = V'$, but from the definition of // it is clear that $\langle V \mid^+ \gamma\rangle \mathbin{/\!/} \langle V \mid^+ \gamma\rangle$. Thus, H2 holds. Now if $V\,\mathbb{V}\,(G)\,V'$, because $\gamma\,\mathbb{S}\,(G)\,\gamma$ we also have $\langle V \mid^+ \gamma\rangle \mathbin{/\!/} \langle V' \mid^+ \gamma\rangle$. But because G is ground, there can be no redex inside V or V'. Thus, by definition of //, $V = V'$.

Now, let's proceed to the induction. It is clear that if $V\,\mathbb{V}\,(\mathbb{1})\,V'$ then $V = V' = *$. Suppose that H3 holds for G_1 and G_2: if $G = G_1 \otimes G_2$ and $V\,\|G\|\,V'$, then $V = (U, W)$ and $V' = (U', W')$ with $U\,\mathbb{V}\,(G_1)\,U'$ and $W\,\mathbb{V}\,(G_2)\,W'$. Hence, by H3, $V = V'$. The same works for $\Box G_1$ or $G_1 \oplus G_2$. □

B.4. Proof that $\simeq$ is an equivalence relation (lemma 13)

Proof. $/\!/$ is clearly symmetric, and it is transitive because the V in the definition of $/\!/$ is unique. It is also reflexive; in order to prove this, we use the characterization of closed normal forms of modal type lemma 11.

First, $\|A\|$, $\mathbb{V}(A)$, $\mathbb{S}(A)$, $\mathbb{T}(A)$ and $\mathbb{E}(A)$ are reflexive. Indeed, it is the case for all these relations except $\|A\|$ by adequacy. Now, for $\|A\|$, we can prove it easily by induction. If $A = B \otimes C$ and $V : A$ in this context, then V is a pair (U, W) because there is no variable in this context, we know that $U\,\mathbb{V}(B)\,U'$ and $W\,\mathbb{V}(C)\,W'$ so $V\,\|A\|\,V'$. If $A = \neg B$ and $S : A$, the only covariables in this context is $\gamma : G$ and G is positive, so $S \neq \gamma$. Thus, $S = [V]$ for some $V : B$. We know that $V\,\mathbb{V}(B)\,V$, thus $S\,\|A\|\,S$.

It is clear that all these relations are symmetric.

Let's show that it is transitive. We will proceed by induction on A. First, if $\|A\|$ is transitive, so are $\mathbb{V}(A)$, $\mathbb{T}(A)$, $\mathbb{S}(A)$ and $\mathbb{E}(A)$. Indeed, let us handle the case where A is positive. In this case, if $\mathbb{S}(A) = \mathbb{E}(A)$ and if $\mathbb{T}(A)$ is positive, then it is also the case of $\mathbb{V}(A)$. Now, suppose that $S_1\,\mathbb{S}(A)\,S_2\,\mathbb{S}(A)\,S_3$, and let $V\,\|A\|\,V'$. We want to show that $\langle V \mid^+ S_1\rangle\,/\!/\langle V' \mid^+ S_2\rangle$. But $\|A\|$ is reflexive, thus $V\,\mathbb{V}(A)\,V$, so we have $\langle V \mid^+ S_1\rangle\,/\!/\langle V \mid^+ S_2\rangle$ and $\langle V \mid^+ S_2\rangle\,/\!/\langle V' \mid^+ S_2\rangle$. $/\!/$ is transitive, so indeed, $S_1\,\mathbb{S}(A)\,S_3$. The same proof works in order to show that $\mathbb{T}(A)$ is transitive.

Now, for the induction, let us treat, for example, the case of $\parr$. If $\mathbb{S}(A)$ and $\mathbb{S}(B)$ are transitive, then $\|A \parr B\|$ is transitive: indeed, if $(S,T)\|A \parr B\|(S',T')\|A \parr B\|(S'',T'')$ then $S\mathbb{S}(A)S'\mathbb{S}(A)S''$ and $T\,\mathbb{S}(B)\,T'\,\mathbb{S}(B)\,T''$, thus $S\,\mathbb{S}(A)\,S''$ and $T\,\mathbb{S}(B)\,T''$, hence $(S,T)\,\|A \parr B\|\,(S'',T'')$. □

B.5. Proof of compatibility with $=_{\beta\eta}$ (definition 8)

Proof. It is enough to show that if $t \to_{\beta\eta} u$ then $t \simeq u$, and similarly for commands, etc.

We will restrict ourselves to reduction of redexes in the proof, so let's convince ourselves that it is possible by dealing with one case. We proceed by induction on the proof of reduction. Suppose that $V \to V'$ and that $\Gamma \mid \Theta(V, W) : A \otimes B; \Delta$. By the inversion lemma (lemma 6), we have proofs $\Gamma_1 \mid \Theta_1 \vdash V : A; \Delta_1$ and $\Gamma_2 \mid \Theta_2 \vdash W : B; \Delta_2$ and a renaming $\theta \in \mathfrak{R}(\Gamma_1, \Gamma_2 \mid \Theta_1, \Theta_2 \vdash \Delta_1, \Delta_2 \Rightarrow \Gamma \mid \Theta \vdash \Delta)$. Also, by subject reduction, we have $\Gamma_1 \mid \Theta_1 \vdash V' : A; \Delta_1$. Thus, by induction $\Gamma_1 \mid \Theta_1 \vdash V \simeq V' : A; \Delta_1$. Moreover, $\simeq$ is reflexive so $W \simeq W$, and by compatibility with typing we get $\Gamma \mid \Theta \vdash (V, W) \simeq (V', W) : A \otimes B; \Delta$.

There exists two kinds of redexes : β-redexes and η-redexes. We will give an example for each of them. For β-redexes, consider for example $\langle \mu\alpha^-.c \mid^- S\rangle : (\Gamma \mid \Theta \vdash \Delta)$.

We want to show that $\langle \mu\alpha^-.c \mid^- S\rangle \simeq c[\alpha := S] : (\Gamma \mid \Theta \vdash \Delta)$. Let $\sigma\,[\![\Gamma \mid \Theta \vdash \Delta]\!]\,\sigma'$. We thus want to show that $\sigma(\langle \mu\alpha^-.c \mid^- S\rangle)\,/\!/\,\sigma(c[\alpha := S])$. But $\sigma(\langle \mu\alpha^-.c \mid^- S\rangle) \to \sigma(c[\alpha := S])$ and $/\!/$ is reflexive, thus indeed $\sigma(\langle \mu\alpha^-.c \mid^- S\rangle)\,/\!/\,\sigma(c[\alpha := S])$.

On the other hand, for η-redexes, we have to deal with the relations $\mathbb{V}(A)$, $\mathbb{T}(A)$, etc. One can see that η-redexes are defined pattern-matching constructs (that is, one that start with $\tilde{\mu}$ or μ) and such that the relations considered are always defined by orthogonality. This means that all cases are dealt with as, for example, the case of $\parr$. Consider $\Gamma \mid \Theta \vdash \mu(\alpha, \beta).\langle V \mid^- (\alpha, \beta)\rangle \simeq V : A \parr B; \Delta$. Let $\sigma\,[\![\Gamma \mid \Theta \vdash \Delta]\!]\,\sigma'$. We want to show that $\sigma(\mu(\alpha, \beta).\langle V \mid^- (\alpha, \beta)\rangle)\mathbb{S}(A \parr B)^{\perp^-}\,\sigma'(V)$. Let $S\,\|A \parr B\|\,S'$; then we can write $S = (T, U)$ and $S' = (T', U')$ with $T\mathbb{S}(A)T'$ and $U\,\mathbb{S}(B)\,U'$.

We want to show that $\langle \sigma(\mu(\alpha,\beta).\langle V \mid^- (\alpha,\beta)\rangle) \mid^- (T,U)\rangle \mathbin{/\!/} \langle \sigma'(V) \mid^- (T',U')\rangle$. The left side reduces to $\langle \sigma(V) \mid^- (T,U)\rangle$, so by saturation, we have to prove that $\langle \sigma(V) \mid^- (T,U)\rangle \mathbin{/\!/} \langle \sigma'(V) \mid^- (T',U')\rangle$. But $V \simeq V$ so $\sigma(V)\ \mathbb{V}\,(A \parr B)\ \sigma'(V)$; and of course $(T,U)\ \mathbb{S}\,(A \parr B)\ (T',U')$, thus indeed $\langle \sigma(V) \mid^- (T,U)\rangle \mathbin{/\!/} \langle \sigma'(V) \mid^- (T',U')\rangle$. This concludes the proof. □

C. Appendix: Conservativity

Proof of theorem 2. First, one can suppose that all values, commands and stacks appearing as subterms are focused, by lemma 14.

Then, we can proceed by induction on the focused proof.

If V is a variable, then it is obvious. Suppose now that V is not a variable.

If $V : \Box(\neg [\![A]\!] \parr [\![B]\!])$ is of the form $V = \Box V'$, and V is not a variable, then by focusing, $V' = \mu\gamma^-.c$ for some c. Because of the $\Box$, V' is typed in an environment without covariables, so c is typed with an environment with only one covariable, $\gamma : \neg [\![A]\!] \parr [\![B]\!]$. We can apply the inversions in any order, so we may as well suppose that the $\parr$ and the $\neg$ has been inverted first and that $c = \langle \mu(\alpha,\beta).\langle \mu[a].c' \mid^- \alpha\rangle \mid^- \gamma\rangle$ with $c' : (\Gamma, a : [\![A]\!] \mid \Theta \vdash \beta : [\![B]\!])$. Then we can apply the induction hypothesis on c'. c can be obtained from c' by appling the left rule of $\to$.

The same goes for $\Box V' : \Box\neg [\![A]\!]$.

If $V : [\![A]\!] \otimes [\![B]\!]$ is a pair (U,W), then by the inversion lemma (lemma 6), U and W can be typed in the same context, so it works. If $V : [\![A]\!] \oplus [\![B]\!]$ it is obvious.

If $t : [\![A]\!]$ is a value we proceed by an obvious induction ; if $t = \mu\alpha^+.c$ then c is typed in a context with only $\alpha : [\![A]\!]$ on the right of the $\vdash$, so it also works.

It is straightforward for stacks of modal types or for stacks of the form $[V] : \neg [\![A]\!]$.

Now, let us look at $[\![\Gamma]\!] \mid \Theta; (S,T) : \neg [\![A]\!] \parr [\![B]\!] \vdash \gamma : [\![C]\!]^?$. S cannot be a covariable, because the only one that may be in the environment is $\gamma : [\![C]\!]$, and $[\![C]\!] \neq \neg [\![A]\!]$. Thus, $S = [V]$ for some V. But V is of modal type, so it can always be typed without γ, if it is present. Thus, we do not contract γ in the derivation of (S,T), and there is always at most one covariables.

Now, for commands: it is clear that the inversion phase do not increase the number of covariables, except for the $\parr$ rule. But then, because the order of the inversions does not matter, we can suppose that every inversion of $\neg [\![A]\!] \parr [\![B]\!]$ is followed by an inversion of the $\neg$, and this corresponds to applying the extra rule for $\to$, so we are safe.

For the intermediate phase, the only rule which may increases the number of or duplicates covariable is the cut rule for $\langle V \mid^+ \alpha\rangle$, on a positive type; for the other rules, simply apply the induction hypothesis. For this rule, we cab show by the subformula property that all positive types involved are modal: thus V can be typed without refering to α. Then, we only have to apply the induction hypothesis. □

$$\frac{c : (\Gamma, x : A, y : B, \Gamma' \mid \Theta \vdash_{inv} \Delta)}{\langle z \mid^{+} \tilde{\mu}(x,y).c\rangle : (\Gamma, z : A \otimes B, \Gamma' \mid \Theta \vdash_{inv} \Delta)} \qquad \frac{c : (\Gamma, x : A_{\square}, \Gamma' \mid \Theta \vdash_{inv} \Delta) \quad c' : (\Gamma, y : B_{\square}, \Gamma' \mid \Theta \vdash_{inv} \Delta)}{\langle z \mid^{+} \tilde{\mu}(\iota_1\ x.c \mid \iota_2\ y.c')\rangle : (\Gamma, z : A_{\square} \oplus B_{\square}, \Gamma' \mid \Theta \vdash_{inv} \Delta)}$$

$$\frac{c : (\Gamma \mid \Theta, x : A_{\square}, y : B_{\square}, \Theta' \vdash_{inv} \Delta)}{\langle z \mid^{+} \tilde{\mu}(a,b).\langle a \mid^{+} \mu\square x.\langle b \mid^{+} \mu\square y.c\rangle\rangle\rangle : (\Gamma \mid \Theta, z : A_{\square} \otimes B_{\square}, \Theta' \vdash_{inv} \Delta)}$$

$$\frac{c : (\Gamma \mid \Theta, x : A, \Theta' \vdash_{inv} \Delta) \quad c' : (\Gamma \mid \Theta, y : B, \Theta' \vdash_{inv} \Delta)}{\langle z \mid^{+} \tilde{\mu}(\iota_1\ x.\langle x \mid^{+} \mu\square a.c\rangle \mid \iota_2\ y.\langle y \mid^{+} \mu\square b.c\rangle)\rangle : (\Gamma \mid \Theta, z : A \oplus B, \Theta' \vdash_{inv} \Delta)}$$

$$\frac{c : (\Gamma, x : \mathbb{1}, \Gamma' \mid \Theta \vdash_{inv} \Delta)}{\langle * \mid^{+} \mu x^{+}.c\rangle : (\Gamma, \Gamma' \mid \Theta \vdash_{inv} \Delta)} \qquad \frac{c : (\Gamma \mid \Theta, x : \mathbb{1}, \Theta' \vdash_{inv} \Delta)}{\langle * \mid^{+} \mu x^{+}.c\rangle : (\Gamma \mid \Theta, \Theta' \vdash_{inv} \Delta)}$$

$$\frac{c : (\Gamma \mid \Theta, x : A, \Theta')}{\langle z \mid^{+} \tilde{\mu}\square x^{+}.c\rangle : (\Gamma, z : \square A \mid \Theta, \Theta')} \qquad \frac{c : (\Gamma, x : \square A, \Gamma' \mid \Theta \vdash_{inv} \Delta)}{\langle \square z \mid^{+} \tilde{\mu} x^{+}.c\rangle : (\Gamma, \Gamma' \mid \Theta, z : A)}$$

$$\frac{c : (\Gamma \mid \Theta \vdash_{inv} \Delta, \alpha : A, \beta : B, \Delta')}{\langle \mu(\alpha,\beta).c \mid^{-} \gamma\rangle : (\Gamma \mid \Theta \vdash_{inv} \Delta, \gamma : A \parr B, \Delta')} \qquad \frac{c : (\Gamma, x : A, \Gamma' \mid \Theta \vdash_{inv} \Delta)}{\langle \mu[x].c \mid^{-} \gamma\rangle : (\Gamma \mid \Theta \vdash_{inv} \Delta, \gamma : \neg A)}$$

$$\frac{(\Gamma \mid \Theta \vdash \Delta) \text{ is inverted} \quad c : (\Gamma \mid \Theta \vdash_{int} \Delta)}{c : (\Gamma \mid \Theta \vdash_{inv} \Delta)}$$

(a) Inversion phase

A sequent $(\Gamma \mid \Theta \vdash \Delta)$ is inverted if every formula in Γ is negative, every formula in Δ is positive and every formula in Θ is non-modal.
In the premises of these rules, we will always suppose that the sequent is inverted.

$$\frac{\Gamma, x : A_{-}, \Gamma' \mid \Theta; S : A_{-} \vdash_{foc} \Delta}{\langle x \mid^{-} S\rangle : (\Gamma, x : A_{-}, \Gamma' \mid \Theta \vdash_{int} \Delta)} \qquad \frac{\Gamma \mid \Theta \vdash_{foc} V : A_{+}; \Delta, \alpha : A_{+}, \Delta'}{\langle V \mid^{+} \alpha\rangle : (\Gamma \mid \Theta \vdash_{int} \Delta, \alpha : A_{+}, \Delta')}$$

$$\frac{\Gamma \mid \Theta, x : A_{-}, \Theta'; S : A_{-} \vdash_{foc} \Delta}{\langle x \mid^{-} S\rangle : (\Gamma \mid \Theta, x : A_{-}, \Theta' \vdash_{int} \Delta)} \qquad \frac{c : (\Gamma, x : A_{\not\square}, y : B_{\not\square} \mid \Theta, z : A_{\not\square} \otimes B_{\not\square}, \Theta' \vdash_{inv} \Delta)}{\langle z \mid^{+} \tilde{\mu}(x,y).c\rangle : (\Gamma \mid \Theta, z : A_{\not\square} \otimes B_{\not\square}, \Theta' \vdash_{int} \Delta)}$$

$$\frac{c : (\Gamma, x : A_{\not\square} \mid \Theta, z : A_{\not\square} \oplus B_{\not\square}, \Theta' \vdash_{inv} \Delta) \quad c' : (\Gamma, y : B_{\not\square} \mid \Theta, z : A_{\not\square} \oplus B_{\not\square}, \Theta' \vdash_{inv} \Delta)}{\langle z \mid^{+} \tilde{\mu}(\iota_1\ x.c \mid \iota_2\ y.c')\rangle : (\Gamma \mid \Theta, z : A_{\not\square} \oplus B_{\not\square}, \Theta' \vdash_{int} \Delta)}$$

(b) Intermediate phase

Here again, $(\Gamma \mid \Theta \vdash \Delta)$ is inverted.

$$\frac{}{\Gamma \mid \Theta, x : A_{\not\square}, \Theta' \vdash_{foc} x : A; \Delta} \qquad \frac{\Gamma^{\square} \mid \Theta \vdash_{foc} V : A; \cdot}{\Gamma \mid \Theta \vdash_{foc} \square V : \square A; \Delta}$$

$$\frac{\Gamma \mid \Theta \vdash_{foc} V : A; \Delta \quad \Gamma \mid \Theta \vdash_{foc} W : B; \Delta}{\Gamma \mid \Theta \vdash_{foc} (V, W) : A \otimes B; \Delta} \qquad \frac{\Gamma \mid \Theta; S : A \vdash_{foc} \Delta \quad \Gamma \mid \Theta; T : B \vdash_{foc} \Delta}{\Gamma \mid \Theta; (S, T) : A \parr B \vdash_{foc} \Delta}$$

$$\frac{\Gamma \mid \Theta \vdash_{foc} V : A_i; \Delta}{\Gamma \mid \Theta \vdash_{foc} \iota_i\ V : A_1 \oplus A_2; \Delta} \qquad \frac{\Gamma \mid \Theta \vdash_{foc} V : A; \Delta}{\Gamma \mid \Theta; [V] : \neg A \vdash_{foc} \Delta} \qquad \frac{}{\Gamma \mid \Theta \vdash_{foc} * : \mathbb{1}; \Delta}$$

$$\frac{c : (\Gamma \mid \Theta \vdash_{inv} \Delta, \alpha : A_{-})}{\Gamma \mid \Theta \vdash_{foc} \mu\alpha^{-}.c : A_{-}; \Delta} \qquad \frac{c : (\Gamma, x : A_{+} \mid \Theta \vdash_{inv} \Delta)}{\Gamma \mid \Theta; \tilde{\mu}x^{+}.c : A_{+} \vdash_{foc} \Delta}$$

(c) Focusing rules

Figure 3: Appendix: The focused proof system

D. Appendix: The $\lambda_{1/2}\mathcal{C}$-calculus

We will here present the $\lambda_{1/2}$-calculus of [Con+19] (renamed here $\lambda_{1/2}\mathcal{C}$ for consistency), and show how the $\lambda_{\Box}\mathcal{C}$-calculus is roughly the same calculus, but where we highlight the fact that "being of level 1" means that we restrict the closure of the variable. We change a bit the calculus: for example, we do not use a boolean type, but we add the $\oplus$ connective, we correct some errors in the rules (for example, the rule of introduction of the pair was faulty, as $\Gamma \vdash (V, W) :^2 A^1 \otimes B^1$ was not possible, but it was claimed that if $\Gamma \vdash t :^1 A$ then $\Gamma \vdash t :^2 A$, etc.), and we remove the restriction that function types cannot return types of level 2. Furthermore, we remove level annotations on pairs and variables, and only allows pairs of values and function applications on values. We refer the reader to the article of Cong et al. for further details.

In their article, they present a λ-calculus where types and variables are annotated by a level, which is 1 or 2. Moreover, the typing rules are so that a term of level 1 is also a term of level 2, and a value of level 1 does not has free variables of level 2. Finally, this calculus is classical: there is $\mathcal{C}$ operator which implements double negation elimination for the level 2 negation. Their calculus is to serve as an intermediate language for compilation; when compiling "intuitionistic" languages (i.e. without control operators, where everything s at level 1), it is more efficient to compile them using control operators and CPS translations that can be compiled as jumps, even though these are "classical" features. Moreover, their CPS translation is not strictly intuitionistic, in the sense that continuations may be discarded or duplicated, but they keep nice properties from the point of view of compilation: second-class closures as continuations can be allocated on the stack, allowing an efficient memory use. To keep this property, they also forbid functions to return second-class elements, which poses no problem to the expressivity of the first-class fragment because second-class elements are only introduced by the compilator. However, even if the calculus is not strictly intuitionistic, the stackability property ensures that once a continuation has been called, continuations that have been allocated after it won't be called, and that when calling a first-class function, all continuations can be deallocated.

I investigated whether this property was interesting from the logical point of view, and found nothing. However, the distinction between two levels was interesting. Indeed, we can see types of level 1 as types of level 2 with an additional restriction on their closure, so that they contain only level 1 variables, and especially, no continuation variables. This leads to the calculus presented in this document: the $\Box$ modality represents the restriction of the closure. In the following parts of this section, I will briefly present the calculus of [Con+19], and show how one can see the $\lambda_{\Box}\mathcal{C}$-calculus as a refinement of their calculus; indeed, because of section D.4, we can see that the $\lambda_{\Box}\mathcal{C}$-calculus has at least the same expressive power as the $\lambda_{1/2}\mathcal{C}$-calculus, and from section D.5 we can see that the CPS translation of the $\lambda_{\Box}\mathcal{C}$-calculus is "the same" as the one of the $\lambda_{1/2}\mathcal{C}$-calculus; however, the $\lambda_{\Box}\mathcal{C}$-calculus is more explicit and its types are nicer than those of the $\lambda_{1/2}\mathcal{C}$-calculus and its level system.

D.1. Grammar and type system

The terms and the reductions are the same as in the $\lambda_{\Box}\mathcal{C}$-calculus, without the $\Box$ connector.

There are two levels: 1 or 2. They are denoted by n, m, k or l.

The types, denoted $A, B, \ldots$, are $\mathbb{1}$, $A^n \otimes B^m$, $A^n \oplus B^m$, $A^n \to B^m$ and $\neg A^n$.

The sequent are of the form $\Gamma \vdash t :^n A$ or $\Gamma \vdash t : \bot$ with $\Gamma = x_1 :^{n_1} A_1, \ldots, x_i :^{n_i} A_i$.

$$\frac{n \leq m}{x :^n A \vdash x :^m A} \qquad \frac{\Gamma \vdash t :^2 \neg(\neg A^n)^2}{\Gamma \vdash \mathcal{C}\ t :^n A}$$

$$\frac{\Gamma \vdash t :^n A \qquad \Gamma', x :^n A \vdash u :^m B}{\Gamma, \Gamma' \vdash \text{let } x = t \text{ in } u :^m B} \qquad \frac{\Gamma \vdash t :^n A \qquad \Gamma', x :^n A \vdash u : \bot}{\Gamma, \Gamma' \vdash \text{let } x = t \text{ in}_\bot u : \bot}$$

$$\frac{}{\Gamma \vdash * :^n \mathbb{1}} \qquad \frac{\Gamma \vdash t :^n \mathbb{1} \qquad \Gamma', x :^n \mathbb{1} \vdash u :^m C}{\Gamma, \Gamma' \vdash \text{let } x = t \text{ in } u :^m C} \qquad \frac{\Gamma \vdash t :^n \mathbb{1} \qquad \Gamma', x :^n \mathbb{1} \vdash u : \bot}{\Gamma, \Gamma' \vdash \text{let } x = t \text{ in}_\bot u : \bot}$$

$$\frac{\Gamma^{\leq n}, x :^m A \vdash t :^k B}{\Gamma \vdash \lambda x.t :^n A^m \to B^k} \qquad \frac{\Gamma \vdash t :^2 A^m \to B^k \qquad \Gamma' \vdash V :^m A \qquad k \leq l}{\Gamma, \Gamma' \vdash t\ V : B^l}$$

$$\frac{\Gamma^{\leq n}, x :^m A \vdash t : \bot}{\Gamma \vdash \lambda^\bot x.t :^n \neg A^m} \qquad \frac{\Gamma \vdash t :^2 \neg A^m \qquad \Gamma' \vdash V :^m A}{\Gamma, \Gamma' \vdash t^\bot\ V : \bot}$$

$$\frac{\Gamma \vdash V :^n A \qquad \Gamma' \vdash W :^m A \qquad \max(n, m) \leq k}{\Gamma, \Gamma' \vdash (V, W) :^k A^n \otimes B^m} \qquad \frac{\Gamma \vdash V :^{n_i} A_i \qquad \max(n_1, n_2) \leq k}{\Gamma \vdash \iota_i\ V :^k A_1 \oplus A_2}$$

$$\frac{\Gamma \vdash t :^k A^n \otimes B^m \qquad \Gamma', x :^n A, y :^m B \vdash u :^l C}{\Gamma, \Gamma' \vdash \text{let } (x, y) = t \text{ in } u :^l C} \qquad \frac{\Gamma \vdash t :^k A^n \otimes B^m \qquad \Gamma', x :^n A, y :^m B \vdash u : \bot}{\Gamma, \Gamma' \vdash \text{let } (x, y) = t \text{ in}_\bot u : \bot}$$

$$\frac{\Gamma \vdash t :^k A^n \oplus B^m \qquad \Gamma', x :^n A \vdash u :^l C \qquad \Gamma', y :^m B \vdash v :^l C}{\Gamma, \Gamma' \vdash \text{match } t \text{ with } \{\iota_1\ x.u \mid \iota_2\ y.v\} :^l C}$$

$$\frac{\Gamma \vdash t :^k A^n \oplus B^m \qquad \Gamma', x :^n A \vdash u : \bot \qquad \Gamma', y :^m B \vdash v : \bot}{\Gamma, \Gamma' \vdash \text{match } t \text{ with}_\bot \{\iota_1\ x.u \mid \iota_2\ y.v\} : \bot}$$

$$\theta \text{ renaming} \frac{\Gamma \vdash t :^n A}{\Gamma' \vdash \theta(t) :^n A} \qquad \frac{\Gamma \vdash t : \bot}{\Gamma' \vdash \theta(t) : \bot}$$

D.2. Translation to the $\lambda_\Box \mathcal{C}$-calculus

We define $[\![A]\!](n)$, the translation of a type A at level n, as follows.

- $[\![A]\!](1) = \Box\,[\![A]\!](2)$
- $[\![\mathbb{1}]\!](2) = \mathbb{1}$
- $[\![A^n \otimes B^m]\!](2) = [\![A]\!](n) \otimes [\![B]\!](m)$
- $[\![A^n \oplus B^m]\!](2) = [\![A]\!](n) \oplus [\![B]\!](m)$
- $[\![A^n \to B^m]\!](2) = [\![A]\!](n) \to [\![B]\!](m)$
- $[\![\neg A^n]\!](2) = \neg\,[\![A]\!](n)$

We could devise a more finetuned translation which would, for example, have $[\![\mathbb{1}]\!](n) = \mathbb{1}$ and $[\![A^1 \otimes B^1]\!](n) = [\![A]\!](1) \otimes [\![B]\!](1)$, but then, given a $\mathbb{1}$ or a $A \otimes B$ with A and B not starting with $\Box$, we would not know if we had to translate it back at level 1 or 2.

$[\![\Gamma]\!]$ is defined by $[\![x :^n A]\!] = x : [\![A]\!](n)$. If $\Gamma \vdash t :^n A$ or $\Gamma \vdash t : \bot$ then $[\![\Gamma]\!] \nmid \cdot \vdash [\![t]\!] : [\![A]\!](n)$ and $[\![\Gamma]\!] \vdash [\![t]\!] : \bot$ (the translation of t here depends on the sequent, $[\![t]\!]$ is an abuse of notation).

Let $\mathrm{extract}(t) = \mathrm{let}\ \Box x = t\ \mathrm{in}\ x$; if $t : \Box A$ then $\mathrm{extract}(t) : A$.

We define the following translation on sequents; double bars means that some proof steps were omitted:

$$\left[\!\!\left[\dfrac{}{x :^n A \vdash x :^n A}\right]\!\!\right] = \dfrac{}{x : [\![A]\!](n) \nmid \cdot \vdash x : [\![A]\!](n)} \qquad \left[\!\!\left[\dfrac{}{x :^1 A \vdash x :^2 A}\right]\!\!\right] = \dfrac{\dfrac{}{x : [\![A]\!](1) \vdash \mathrm{extract}(x) : [\![A]\!](2)}}{}$$

$$\left[\!\!\left[\dfrac{\Gamma \vdash V :^n A \qquad \Gamma' \vdash W :^m B}{\Gamma, \Gamma' \vdash (V, W) :^2 A^n \otimes B^m}\right]\!\!\right] = \dfrac{\Gamma \nmid \cdot \vdash [\![V]\!] : [\![A]\!](n) \qquad \Gamma' \nmid \cdot \vdash [\![W]\!] : [\![B]\!](m)}{\dfrac{}{\Gamma, \Gamma' \nmid \cdot \vdash \mathrm{let}\ x = [\![V]\!]\ \mathrm{in}\ \mathrm{let}\ y = [\![W]\!]\ \mathrm{in}\ (x, y) : [\![A]\!](n) \otimes [\![B]\!](m)}}$$

$$\left[\!\!\left[\dfrac{\Gamma \vdash V :^1 A \qquad \Gamma' \vdash W :^1 B}{\Gamma, \Gamma' \vdash (V, W) :^2 A^1 \otimes B^1}\right]\!\!\right] = \dfrac{\Gamma \nmid \cdot \vdash [\![V]\!] : [\![A]\!](1) \qquad \Gamma' \nmid \cdot \vdash [\![W]\!] : [\![B]\!](1)}{\dfrac{}{\Gamma, \Gamma' \nmid \cdot \vdash \mathrm{let}\ x = [\![V]\!]\ \mathrm{in}\ \mathrm{let}\ y = [\![W]\!]\ \mathrm{in}\ \Box(x, y) : \Box([\![A]\!](1) \otimes [\![B]\!](1))}}$$

This is valid because $\varpi([\![A]\!](1)) = \varpi([\![B]\!](1)) = \Box$.

$$\left[\!\!\left[\dfrac{\Gamma \vdash t :^2 A^n \otimes B^m \qquad \Gamma', x :^n, y :^m \vdash u :^k C}{\Gamma, \Gamma' \vdash \mathrm{let}\ (x, y) = t\ \mathrm{in}\ u}\right]\!\!\right] = \dfrac{[\![\Gamma]\!] \nmid \cdot \vdash [\![t]\!] : [\![A]\!](n) \otimes [\![B]\!](m) \qquad [\![\Gamma']\!], x : [\![A]\!](n), y : [\![B]\!](m) \nmid \cdot \vdash [\![u]\!] : [\![C]\!](k)}{[\![\Gamma]\!], [\![\Gamma']\!] \nmid \cdot \vdash \mathrm{let}\ (x, y) = t\ \mathrm{in}\ u : [\![C]\!](k)}$$

$$\left[\!\!\left[\dfrac{\Gamma \vdash t :^1 A^n \otimes B^m \qquad \Gamma', x :^n, y :^m \vdash u :^k C}{\Gamma, \Gamma' \vdash \mathrm{let}\ (x, y) = t\ \mathrm{in}\ u}\right]\!\!\right] = \dfrac{\dfrac{[\![\Gamma]\!] \nmid \cdot \vdash [\![t]\!] : \Box([\![A]\!](n) \otimes [\![B]\!](m))}{[\![\Gamma]\!] \nmid \cdot \vdash \mathrm{extract}([\![t]\!]) : [\![A]\!](n) \otimes [\![B]\!](m)} \qquad [\![\Gamma']\!], x : [\![A]\!](n), y : [\![B]\!](m) \nmid \cdot \vdash [\![u]\!] : [\![C]\!](k)}{[\![\Gamma]\!], [\![\Gamma']\!] \nmid \cdot \vdash \mathrm{let}\ (x, y) = \mathrm{extract}(t)\ \mathrm{in}\ u : [\![C]\!](k)}$$

$$\left[\!\!\left[\frac{\Gamma^{\leq 1}, x :^n A \vdash t :^m B}{\Gamma \vdash \lambda x.t :^1 A^n \to B^m}\right]\!\!\right] =$$

$$\frac{\dfrac{\dfrac{\left[\!\left[\Gamma^{\leq 1}, x : [\![A]\!]\,(n) \mid \cdot \vdash [\![t]\!] : [\![B]\!]\,(m)\right]\!\right]}{\left[\!\left[\Gamma^{\leq 1}\right]\!\right] \mid \cdot \vdash \lambda x.t : [\![A]\!]\,(n) \to [\![B]\!]\,(m)}}{\left[\!\left[\Gamma^{\leq 1}\right]\!\right] \mid \cdot \vdash \Box\lambda x.\,[\![t]\!] : \Box([\![A]\!]\,(n) \to [\![B]\!]\,(m))}}{[\![\Gamma]\!] \mid \cdot \vdash \Box\lambda x.\,[\![t]\!] : \Box([\![A]\!]\,(n) \to [\![B]\!]\,(m))}$$

This works because $\left[\!\left[\Gamma^{\leq 1}\right]\!\right] \subseteq [\![\Gamma]\!]^{\Box}$.

$$\left[\!\!\left[\frac{\Gamma \vdash t :^1 A^n \to B^1 \qquad \Gamma' \vdash V :^n A}{\Gamma, \Gamma' \vdash t\ u : B^2}\right]\!\!\right] =$$

$$\frac{\dfrac{\dfrac{[\![\Gamma]\!] \mid \cdot \vdash [\![t]\!] : \Box([\![A]\!]\,(n) \to [\![B]\!]\,(1))}{[\![\Gamma]\!] \mid \cdot \vdash \mathrm{extract}([\![t]\!]) : [\![A]\!]\,(n) \to [\![B]\!]\,(1)} \qquad [\![\Gamma']\!] \mid \cdot \vdash [\![V]\!] : [\![A]\!]\,(n)}{[\![\Gamma]\!], [\![\Gamma]\!] \mid \cdot \vdash \mathrm{let}\ f = \mathrm{extract}\,[\![t]\!]\ \mathrm{in}\ \mathrm{let}\ x = [\![V]\!]\ \mathrm{in}\ f\ x : [\![B]\!]\,(1)}}{[\![\Gamma]\!], [\![\Gamma]\!] \mid \cdot \vdash \mathrm{extract}(\mathrm{let}\ f = \mathrm{extract}\,[\![t]\!]\ \mathrm{in}\ \mathrm{let}\ x = [\![V]\!]\ \mathrm{in}\ f\ x) : [\![B]\!]\,(2)}$$

$$\left[\!\!\left[\frac{\Gamma \vdash t :^2 \neg(\neg A^n)^2}{\Gamma \vdash \mathcal{C}\ t :^n A}\right]\!\!\right] = \frac{[\![\Gamma]\!] \mid \cdot \vdash [\![t]\!] : \neg\neg [\![A]\!]\,(n)}{[\![\Gamma]\!] \mid \cdot \vdash \mathcal{C}\ \ [\![t]\!] : [\![A]\!]\,(n)}$$

The translation of other cases follow the same idea, adding extract, let and $\Box$ when necessary.

This translation does not preserve values, so it does not preserve substitutions and reductions. However, it is so only because it add some let here and there, so the translation is still acceptable.

D.3. Translation from the $\lambda_{\Box}\mathcal{C}$-calculus

We define $n(A)$ which represents the level at which the type A should be translated as $n(A) = 1$ if $\varpi(A) = \Box$ and $n(A) = 2$ else. Essentially, a modal type is translated at level 1 and a non-modal one at level 2.

and $[\![A]\!]$, the translation:

- $[\![\mathbb{1}]\!] = \mathbb{1}$
- $[\![A \oplus B]\!] = [\![A]\!]^{n(A)} \oplus [\![B]\!]^{n(B)}$
- $[\![A \otimes B]\!] = [\![A]\!]^{n(A)} \otimes [\![B]\!]^{n(B)}$
- $[\![A \to B]\!] = [\![A]\!]^{n(A)} \to [\![B]\!]^{n(B)}$
- $[\![\neg A]\!] = \neg\, [\![A]\!]^{n(A)}$
- $[\![\Box A]\!] = [\![A]\!]$

$[\![\Gamma]\!]$ is defined by $[\![\cdot]\!] = \cdot$ and $[\![\Gamma, x : A]\!] = [\![\Gamma]\!], x :^{n(A)} [\![A]\!]$. $[\![\Theta]\!]^{\Box}$ is defined by $[\![\cdot]\!]^{\Box} = \cdot$ and $[\![\Theta, x : A]\!]^{\Box} = [\![\Theta]\!]^{\Box}, x :^1 [\![A]\!]$.

The induction is done solely on the values and terms, not on the derivation:

- $[\![x]\!] = x$
- $[\![*]\!] = *$
- $[\![\iota_i\ V]\!] = \iota_i\ V$
- $[\![(V, W)]\!] = ([\![V]\!], [\![W]\!])$
- $[\![\Box V]\!] = [\![V]\!]$
- $[\![\lambda x.t]\!] = \lambda x.\, [\![t]\!]$
- $[\![\lambda^\perp x.t]\!] = \lambda^\perp x.\, [\![t]\!]$
- $[\![\text{let } x = t \text{ in } u]\!] = \text{let } x = [\![t]\!] \text{ in } [\![u]\!]$
- $[\![\text{let } x = t \text{ in}_\perp u]\!] = \text{let } x = [\![t]\!] \text{ in}_\perp [\![u]\!]$
- $[\![\text{let } \Box x = t \text{ in } u]\!] = \text{let } x = [\![t]\!] \text{ in } [\![u]\!]$
- $[\![\text{let } \Box x = t \text{ in}_\perp u]\!] = \text{let } x = [\![t]\!] \text{ in}_\perp [\![u]\!]$
- $[\![\text{match } t \text{ with } \{\iota_1\ x.u \mid \iota_2\ y.v\}]\!] = \text{match } [\![t]\!] \text{ with } \{\iota_1\ x.\, [\![u]\!] \mid \iota_2\ y.\, [\![v]\!]\}$
- $[\![\text{match } t \text{ with}_\perp \{\iota_1\ x.u \mid \iota_2\ y.v\}]\!] = \text{match } [\![t]\!] \text{ with}_\perp \{\iota_1\ x.\, [\![u]\!] \mid \iota_2\ y.\, [\![v]\!]\}$
- $[\![\text{let } (x, y) = t \text{ in } u]\!] = \text{let } (x, y) = [\![t]\!] \text{ in } [\![u]\!]$
- $[\![\text{let } (x, y) = t \text{ in}_\perp u]\!] = \text{let } (x, y) = [\![t]\!] \text{ in}_\perp [\![u]\!]$
- $[\![t\ V]\!] = [\![t]\!]\ [\![V]\!]$
- $[\![t^\perp\ V]\!] = [\![t]\!]^\perp\ [\![V]\!]$
- $[\![\mathcal{C}\ t]\!] = \mathcal{C}\ [\![t]\!]$

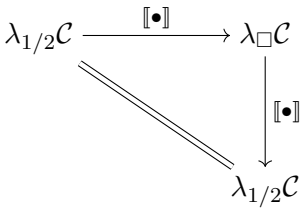
If $t \to u$ then $[\![t]\!] \to^* [\![u]\!]$.

If $\Gamma \nmid \Theta \vdash t : A$ then $[\![\Gamma]\!], [\![\Theta]\!]^\Box \vdash [\![t]\!] :^{n(A)} [\![A]\!]$. If $\Gamma \nmid \Theta \vdash t : \perp$ then $[\![\Gamma]\!], [\![\Theta]\!]^\Box \vdash [\![t]\!] : \perp$.

D.4. Factorization

Here, we see that we lack enough η-rules. However, everything works if we add the rule $\text{let } x = t \text{ in } x \to_\eta t$ in the $\lambda_{1/2}\mathcal{C}$-calculus.

For each type A in $\lambda_{1/2}\mathcal{C}$-calculus, we have that $n([\![A]\!](k)) = k$ and $[\![[\![A]\!](k)]\!] = A$. Thus, by composing the preceding translation, we get that $\Gamma \vdash t :^n A$ then $[\![[\![\Gamma]\!]]\!] \vdash [\![[\![t]\!]]\!] :^n A$. Moreover, $[\![[\![V]\!]]\!] \to V$ for all V, and $[\![\text{extract}(t)]\!] = \text{let } y = t \text{ in } y \to_\eta t$. The rest follows by induction, because all the first traduction does is adding some let and some extract on terms that "should" be values. Thus, the following diagram commutes, up to $\cong$:



D.5. CPS factorization

Clearly, the translation on types corresponds. Now, that means that if $\Gamma \vdash t :^n A$ then $\mathrm{CPS}\,[\![\Gamma]\!], k :^2 \neg\,\mathrm{CPS}\,[\![A]\!]^n \vdash \left[\!\left[[\![t]\!]^\lambda_{\mathsf{t}}\,(k) \right]\!\right] : \bot$ and if $\Gamma \vdash t : \bot$ then $\mathrm{CPS}\,[\![\Gamma]\!] \vdash \left[\!\left[[\![t]\!]^\lambda_{\bot} \right]\!\right] : \bot$. Additionally, we will prove that we also have that $\left[\!\left[[\![t]\!]^\lambda_{\mathsf{t}}\,(M) \right]\!\right] \to^* \mathrm{CPS}\,[\![t]\!]\,([\![M]\!])$.

First, we have $[\![\mathrm{extract}(t)]\!]^\lambda_{\mathsf{t}}\,(M) = [\![\mathrm{let}\ y = t\ \mathrm{in}\ y]\!]^\lambda_{\mathsf{t}}\,(M) = [\![t]\!]^\lambda_{\mathsf{t}}\,(M)\lambda^\bot y.M^\bot\ y \to^* [\![t]\!]^\lambda_{\mathsf{t}}\,(M)$. The rest follows easily, for example:

$$
\begin{aligned}
&\left[\!\left[[\![\mathcal{C}\ t]\!]^\lambda_{\mathsf{t}}\,(M) \right]\!\right] \\
&\qquad = \left[\!\left[[\![\mathcal{C}\ \ [\![t]\!]]\!]^\lambda_{\mathsf{t}}\,(M) \right]\!\right] \\
&\qquad = \left[\!\left[[\![[\![t]\!]]\!]^\lambda_{\mathsf{t}}\,(\lambda^\bot v.v^\bot\ M) \right]\!\right] \\
&\qquad \to^* \mathrm{CPS}\,[\![t]\!]\,(\left[\!\left[\lambda^\bot v.v^\bot\ M \right]\!\right]) \\
&\qquad = \mathrm{CPS}\,[\![t]\!]\,(\lambda^\bot v.v^\bot\ \ [\![M]\!]) \\
&\qquad\qquad = \mathrm{CPS}\,[\![\mathcal{C}\ t]\!]\,([\![M]\!])
\end{aligned}
$$

Or, with $t :^1 (A^n \to B^1)$ and $t\ V :^2 B$:

$$
\begin{aligned}
&\left[\!\left[[\![[\![t\ V]\!]]\!]^\lambda_{\mathsf{t}}\,(M) \right]\!\right] \\
&\qquad = \left[\!\left[[\![\mathrm{extract}(\mathrm{let}\ f = \mathrm{extract}\,[\![t]\!]\ \mathrm{in\ let}\ x = [\![V]\!]\ \mathrm{in}\ f\ x)]\!]^\lambda_{\mathsf{t}}\,(M) \right]\!\right] \\
&\qquad\quad \to^* \left[\!\left[[\![\mathrm{let}\ f = \mathrm{extract}\,[\![t]\!]\ \mathrm{in\ let}\ x = [\![V]\!]\ \mathrm{in}\ f\ x]\!]^\lambda_{\mathsf{t}}\,(M) \right]\!\right] \\
&\qquad \to^* \left[\!\left[[\![\mathrm{extract}\,[\![t]\!]]\!]^\lambda_{\mathsf{t}}\,(\lambda^\bot f.\,[\![[\![V]\!]]\!]^\lambda_{\mathsf{t}}\,(\lambda^\bot x.(\lambda^\bot k.k^\bot\ (x, \lambda^\bot y.M^\bot\ y))^\bot\ f)) \right]\!\right] \\
&\qquad\quad \to^* \left[\!\left[[\![[\![t]\!]]\!]^\lambda_{\mathsf{t}}\,(\lambda^\bot f.\,[\![[\![V]\!]]\!]^\lambda_{\mathsf{t}}\,(\lambda^\bot x.(\lambda^\bot k.k^\bot\ (x, \lambda^\bot y.M^\bot\ y))^\bot\ f)) \right]\!\right] \\
&\qquad \to^* \mathrm{CPS}\,[\![t]\!]\,(\lambda^\bot f.\left[\!\left[[\![[\![V]\!]]\!]^\lambda_{\mathsf{t}}\,(\lambda^\bot x.(\lambda^\bot k.k^\bot\ (x, \lambda^\bot y.M^\bot\ y))^\bot\ f) \right]\!\right]) \\
&\qquad \to^* \mathrm{CPS}\,[\![t]\!]\,(\lambda^\bot f.\,\mathrm{CPS}\,[\![V]\!]\,(\lambda^\bot x.(\lambda^\bot k.k^\bot\ (x, \lambda^\bot y.M^\bot\ y)^\bot\ f))) \\
&\qquad\quad \to^* \mathrm{CPS}\,[\![t]\!]\,(\lambda^\bot f.\,\mathrm{CPS}\,[\![V]\!]\,(\lambda^\bot x.f^\bot\ (x, \lambda^\bot y.M^\bot\ y))) \\
&\qquad\quad = \mathrm{CPS}\,[\![t]\!]\,(\lambda^\bot f.(\lambda^\bot x.f^\bot\ (x, \lambda^\bot y.M^\bot\ y))^\bot\ \ \mathrm{CPS}\,[\![V]\!]_v) \\
&\qquad\qquad = \mathrm{CPS}\,[\![t]\!]\,(\lambda^\bot f.f^\bot\ (\mathrm{CPS}\,[\![V]\!]_v, \lambda^\bot y.M^\bot\ y)) \\
&\qquad\qquad\qquad = \mathrm{CPS}\,[\![t\ V]\!]\,(M)
\end{aligned}
$$

Thus again, up to $\cong$, the following diagram commutes: