CATEGORICAL PROOF THEORY OF CO-INTEUITIONISTIC LINEAR LOGIC

GIANLUIGI BELLIN

Dipartimento di Informatica, Università di Verona, Strada Le Grazie, 37134 Verona Italy
e-mail address: gianluigi.bellin@univr.it

Revision Note. This is a revised and corrected version of the article originally published on September 10, 2014.

Abstract. To provide a categorical semantics for co-intuitionistic logic, one has to face the fact, noted by Tristan Crolard, that the definition of co-exponents as adjuncts of co-products does not work in the category Set, where co-products are disjoint unions. Following the familiar construction of models of intuitionistic linear logic with exponential “!”, we build models of co-intuitionistic logic in symmetric monoidal closed categories with additional structure, using a variant of Crolard’s term assignment to co-intuitionistic logic in the construction of a free category.

Preface

This paper sketches a categorical semantics for co-intuitionistic logic, advancing a line of proof-theoretic research developed in [1, 2, 3, 5, 7]. Co-intuitionistic logic, also called dual-intuitionistic [21, 35, 36], may be superficially regarded as completely determined by the duality, as in its lattice-theoretic semantics. A co-Heyting algebra is a (distributive) lattice C such that its opposite C^op is a Heyting algebra. In a Heyting algebra implication B → A is defined as the right adjoint of meet, so in a co-Heyting algebra C co-implication (or subtraction) A \setminus B is defined as the left adjoint of join:

\[
\begin{align*}
C \land B & \leq A \\
C & \leq B \rightarrow A
\end{align*}
\]

\[
\begin{align*}
A & \leq B \lor C \\
A \setminus B & \leq C
\end{align*}
\]

A bi-Heyting algebra is a lattice that has both the structure of a Heyting and of a co-Heyting algebra. The logic of bi-Heyting algebras was introduced by Cecylia Rauszer [30, 31] (called Heyting-Brouwer logic), who defined also its Kripke semantics; a category-theoretical approach to the topic is due to Makkai, Reyes and Zolfaghari [28, 32]. The suggestion by F. W. Lawvere to use co-Heyting algebras as a logical framework to treat the topological notion of boundary has not been fully explored yet (but see recent work by Pagliani [29]).

2012 ACM CCS: [Theory of computation]: Logic—Proof theory; Linear logic.

Key words and phrases: Categorical Proof Theory, Intuitionistic duality, Categorical Semantics of Intuitionistic Linear Logic, Semantics of coroutines and concurrent computations.
Early research showed that the extension of first order intuitionistic logic with subtraction yields an intermediate logic of constant domains \[20\]. In a rich and interesting paper \[18\] T. Crolard showed, essentially by Joyal’s argument, that Cartesian closed categories with exponents and co-exponents are degenerate; in fact even the topological models of bi-intuitionistic logic, i.e., bi-topological spaces, are degenerate. Crolard’s motivations are mainly computational: he studies bi-intuitionistic logic in the framework of the classical \(\lambda\mu\) calculus, to provide a type-theoretic analysis of the notion of coroutine; then he identifies a subclass of safe coroutines that can be typed constructively \[19\]. From our viewpoint, Crolard’s work suggests two directions of research. On one hand, it opens the way to a “bottom up” approach to safe coroutines, independent of the \(\lambda\mu\) calculus, i.e., co-intuitionistic coroutines \[3, 1, 2, 7\]. On the other hand, the question arises whether the collapse of algebraic and topological models may be avoided by building the intuitionistic and co-intuitionistic sides separately, starting from distinct sets of elementary formulas, and then by joining the two sides with mixed connectives (mainly, two negations expressing the duality): this is our variant of bi-intuitionistic logic, presented in \[5, 1, 6\].

Both of these tasks were advocated by this author and pursued within a project of “logic for pragmatics” with motivations from linguistics and natural language representation \[1, 5, 9\]. In the characterization of the logical properties of “illocutionary acts”, such as asserting, making hypotheses and conjectures one finds in natural reasoning forms of duality that can be related to intuitionistic dualities. For co-intuitionistic logic Crolard’s term assignment has been adapted to a sequent-style natural deduction setting with single-premise and multiple-conclusions. For (our variant of) bi-intuitionistic logic Kripke semantics has been given (both in \(S_4\) and in bi-modal \(S_4\)) and a sequent calculus has been proposed where sequents are of the form

\[\Gamma ; \Rightarrow A ; \Upsilon \quad \text{or} \quad \Gamma ; C \Rightarrow ; \Upsilon\]

where \(\Gamma\) and \(A\) are intuitionistic (assertive) formulas and \(C\) and \(\Upsilon\) co-intuitionistic (hypothetical).

But from the viewpoint of category theory a crucial remark by Crolard shows that already in co-intuitionistic logic there is a problem: namely, only trivial co-exponents exist in the category \(\text{Set}\). Indeed the categorical semantics of intuitionistic disjunction is given by coproducts \[24\], which in \(\text{Set}\) are represented by disjoint unions. On the other hand the categorical semantics of subtraction is given by co-exponents. The co-exponent of \(A\) and \(B\) is an object \(B_A\) together with an arrow \(\exists_{A,B} : B \to B_A \oplus A\) such that for any arrow \(f : B \to C \oplus B\) there exists a unique \(f_* : B_A \to C\) such that the following diagram commutes:

\[
\begin{array}{ccc}
B & \xrightarrow{f} & C \oplus A \\
\downarrow & & \downarrow \\exists_{A,B} \\
B_A \oplus A & \xrightarrow{f_* \oplus id_A} & C \oplus A
\end{array}
\]

It follows that

in the category of sets, the co-exponent \(B_A\) of two sets \(A\) and \(B\) is defined if and only if \(A = \emptyset\) or \(B = \emptyset\) (see \[1N\], Proposition 1.15).

The proof is instructive: in \(\text{Set}\), the coproduct \(\oplus\) is disjoint union; thus if \(A \neq \emptyset \neq B\) then the functions \(f\) and \(\exists_{A,B}\) for every \(b \in B\) must choose a side, left or right, of the coproduct in their target and moreover \(f_* \oplus 1_A\) leaves the side unchanged. Hence, if we take
a nonempty set \( C \) and \( f \) with the property that for some \( b \) different sides are chosen by \( f \) and \( \exists_{A,B} \), then the diagram does not commute.

Thus to have a categorical semantics of co-exponents we need categories where a different notion of disjunction is modelled. The connective \textit{par} of linear logic is a good candidate and a treatment of \textit{par} is available in \textit{full intuitionistic linear logic} (FILL) \cite{16 22}, with a proof-theory and a categorical semantics. The multiple-conclusion consequence relation of FILL and its term assignment have given motivation and inspiration to our work, as a calculus where a distinct term is assigned to each formula in the succedent. The language of FILL has tensor \((\otimes)\), linear implication \((-\circ)\) and \textit{par} \((\wp)\) and a main proof-theoretic concern has been the compatibility between \textit{par} and linear implication, namely, to find restrictions on the introduction of linear implication that guarantee its \textit{functional} intuitionistic character and at the same time allow to prove cut-elimination (on this point see also \cite{4}).

Linear co-intuitionistic logic appears already in Schellinx \cite{33} and Lambek \cite{23}. Works by R. Blute, J. Cockett, R. Sely and T. Trimble on \textit{weakly distributive categories} \cite{14 15} provide a sophisticated technology of natural deduction, proof-nets and categorical models for various systems of linear logic without an involutory negation; Cockett and Seely \cite{17} consider also non-commutative systems with implications and subtractions. When \textit{boxes} or other conditions are given for intuitionistic linear implications and co-intuitionistic subtractions, these systems provide a suitable categorical proof theory of linear bi-intuitionism.

To construct categorical models of linear co-intuitionistic logic it suffices to notice that in \textit{monoidal categories} \textit{par} can be modelled by a monoidal operation and co-exponents as the \textit{left adjoint} of \textit{par}. The main task then is to model Girard’s exponential \textit{why not}?: in this way a categorical semantics for co-intuitionistic logic can be recovered by applying the dual of Girard’s translation of intuitionistic logic into linear logic, namely:

\[
\begin{align*}
(p)^\circ &= p \\
(f)^\circ &= 0 \\
(C \wp D)^\circ &= (C^\circ \oplus D^\circ) = (C^\circ)\wp(D^\circ) \\
(C \setminus D)^\circ &= C^\circ \setminus (D^\circ) \\
(E \vdash C_1, \ldots, C_n)^\circ &= (E^\circ) \vdash (C_1^\circ), \ldots, (C_n^\circ)
\end{align*}
\]

where \textit{0} is the identity of \oplus and we use “\textit{\setminus}” both in linear and in non-linear co-intuitionistic logic.

The task amounts to dualizing Nick Benton, Gavin Bierman, Valeria de Paiva and Martin Hyland’s well-known semantics for intuitionistic linear logic \cite{11}. This may be regarded as a routine exercise, except that one has to provide a term assignment suitable for the purpose. In this task we build on a term assignment to multiplicative co-intuitionistic logic, which has been proposed as an abstract \textit{distributed calculus} dualizing the linear \(\lambda\) calculus \cite{2 3 7}, in our view such a dualization underlies the translation of the linear \(\lambda\)-calculus into the \(\pi\)-calculus (see \cite{10}).

As a matter of fact, Nick Benton’s \textit{mixed Linear and Non-Linear} logic \cite{12} may give us not only an easier approach to modelling the exponentials but also the key to a categorical semantics of (our version of) bi-intuitionistic logic: indeed, by dualizing the linear part of Benton’s system we may obtain both a proof-theoretic and a category theoretic framework for mixed co-intuitionistic linear and intuitionistic logic and thus also for bi-intuitionistic logic - of course, we need to use the exponential \textit{why not} and dualize Girard’s translation.
But then a categorical investigation of linear co-intuitionistic logic and of the why not? exponential is a preliminary step in this direction and has an independent interest.

1. Proof Theory

The language of co-intuitionistic linear logic, given an infinite sequence of elementary formulas \( \eta_1, \eta_2, \ldots \), is defined by the following grammar:

\[
C, D := \eta \mid \bot \mid C \varphi D \mid C \triangleleft D \mid ?C
\]

The rules of sequent-style natural deduction for co-ILL for co-Intuitionistic Linear Logic are given in Table 1. As co-intuitionistic linear logic may be quite unfamiliar, we sketch an intuitive explanation of its proof theory. We think of co-intuitionism as being about making hypotheses [1, 5, 6]. It has a consequence relation of the form

\[
H \vdash H_1, \ldots, H_n.
\]  

(1.1)

Suppose \( H \) is a hypothesis: which (disjunctive sequence of) hypotheses \( H_1 \) or \( \ldots \) or \( H_n \) follow from \( H \)? Since the logic is linear, commas in the meta-theory stand for Girard’s par and the structural rules Weakening and Contraction are not allowed. A relevant feature, which we shall not discuss here, is that the consequence relation may be seen as distributed, i.e., we may think of the alternatives \( H_1, \ldots, H_n \) in (1.1) as lying in different locations [2, 7].

Table 1: Natural Deduction for co-ILL

\[
\begin{array}{cc}
\text{assumption} & \text{substitution} \\
A \vdash A & E \vdash \Gamma, A \\
\bot \vdash \Gamma & A \vdash \Delta \\
E \vdash \Gamma, \bot & \bot \vdash \\
E \vdash \Gamma, C & D \vdash \Delta \\
\rightarrow \vdash E \vdash \Gamma, C \rightarrow D, \Delta \\
H \vdash \Upsilon, C \triangleleft D & C \vdash D, \Delta \\
\varphi \vdash E \\
E \vdash \Gamma, C_0, C_1 & H \vdash \Upsilon, C_0 \varphi C_1 \\
E \vdash \Gamma, C_0 \varphi C_1 & \varphi \vdash \Gamma_0 \\
H \vdash \Upsilon, \Gamma_0, \Gamma_1 \\
\text{dereliction} & \text{storage} \\
E \vdash \Gamma, C & C \vdash ?\Delta \\
E \vdash \Gamma, ?C & H \vdash \Upsilon, ?\Delta \\
\text{weakening} & \text{contraction} \\
E \vdash \Gamma & E \vdash \Gamma, ?C
\end{array}
\]

1In accordance with our interpretation of co-intuitionism as a logic of hypotheses, we may write elementary formulas \( \eta \) as \( \eta p \), where “\( \eta \)” is a sign for the illocutionary force of hypothesis and \( p \) is an atomic proposition. Such a linguistic analysis plays no explicit role in this paper.
The main connectives are subtraction $A \setminus B$ (possibly $A$ and not $B$) and Girard’s $par A \varphi B$. Natural Deduction inference rules for subtraction (in a sequent form) are as follows.

$$
\text{\texttt{\textbackslash{-}intro}} \quad \frac{H \vdash \Gamma, C \quad D \vdash \Delta}{H \vdash \Gamma, C \setminus D, \Delta} \\
\text{\texttt{\textbackslash{-}elim}} \quad \frac{H \vdash \Delta, C \setminus D \quad C \vdash D, \Upsilon}{H \vdash \Delta, \Upsilon}
$$

Notice that in the $\texttt{\textbackslash{-}elim}$ rule the evidence that $D$ may be derivable from $C$ given by the right premise has become inconsistent with the hypothesis $C \setminus D$ in the left premise; in the conclusion we drop $D$ and we set aside the evidence for the inconsistent alternative. Namely, such evidence is not destroyed, but rather stored somewhere for future use.

If the left premise of $\texttt{\textbackslash{-}elim}$, deriving $C \setminus D$ or $\Delta$ from $H$, has been obtained by a $\texttt{\textbackslash{-}intro}$, this inference has the form

$$
\frac{H \vdash \Delta_1, C \quad D \vdash \Delta_2}{H \vdash \Delta_1, \Delta_2, C \setminus D}.
$$

Then the pair of introduction/elimination rules can be eliminated: using the removed evidence that $D$ with $\Upsilon$ are derivable from $C$ (right premise of the $\texttt{\textbackslash{-}elim}$) we can conclude that $\Delta_1, \Delta_2, \Upsilon$ are derivable from $H$. This is, in a nutshell, the principle of normalization (or cut-elimination) for subtraction.

The storage operation is made explicit in the rules for the $\texttt{\textbang}$ operator of linear logic. Here an entire derivation $d$ of $\texttt{\textbang} \Delta$ from $C$ (where $\texttt{\textbang} \Delta = \texttt{\textbang} D_1, \ldots, \texttt{\textbang} D_n$) is set aside; what is accessible now is something like a non-logical axiom of the form $C \vdash \texttt{\textbang} \Delta$. However in the process of normalization the derivation $d$ may be recovered to be used, discarded or copied in the interaction of a storage rule with dereliction, weakening or contraction: all of this is conceptually clear, thanks to J-Y. Girard, and has been mathematically analyzed in the geometry of interaction.

1.1. From Crolard’s classical coroutines to co-intuitionistic ones. Crolard [19] provides a term assignment to the subtraction rules in the framework of Parigot’s $\lambda\mu$-calculus, typed in a sequent-style natural deduction system. The $\lambda\mu$-calculus provides a typing system for functional programs with continuations and a computational interpretation of classical logic (see, e.g., [20] [31]).

In the type system for the $\lambda\mu$ calculus sequents may be written in the form $\Gamma \vdash t : A \mid \Delta$, with contexts $\Gamma = x_1 : C_1, \ldots, x_m : C_m$ and $\Delta = \alpha_1 : D_1, \ldots, \alpha_n : D_n$, where the $x_i$ are variables and the $\alpha_j$ are $\mu$-variables (or co-names). In addition to the rules of the simply typed lambda calculus, there are naming rules

$$
\Gamma \vdash t : A \mid \Delta, \alpha : A, \Delta' \\
\Gamma \vdash [\alpha]t : \bot \mid \alpha : A, \Delta' \quad \Gamma \vdash \mu\alpha.t : A \mid \Delta'^\mu
$$

whose effect is to “change the goal” of a derivation and which allow us to represent the familiar double negation rule in Prawitz Natural Deduction.

Crolard extends the $\lambda\mu$ calculus with introduction and elimination rules for subtraction.

<sup>2</sup>Actually in Crolard [19] the introduction rule is given in the more general form of $\texttt{\textbang\textbackslash{-}intro}$ with two sequent premises (which we use below) and more general continuation contexts occur in place of $\beta$; the above formulation is logically equivalent and sufficient for our purpose.
The reduction of a redex of the form
\[ \Gamma \vdash t : A \mid \Delta \]
\[ \Gamma \vdash \text{make-coroutine}(t, \beta) : A \setminus B \mid \beta : B, \Delta \] \text{I}
\[ \Gamma \vdash t : A \setminus B \mid \Delta \quad \Gamma, x : A \vdash u : B \mid \Delta \]
\[ \Gamma \vdash \text{resume} t \text{ with } x \mapsto u : C \mid \Delta \text{ E} \]

is as follows:
\[ \Gamma \vdash t : A \mid \Delta \quad \Gamma, x : A \vdash u : B \mid \Delta \]
\[ \Gamma \vdash \text{make-coroutine}(t, \beta) : A \setminus B \mid \beta : B, \Delta \]
\[ \Gamma \vdash \text{resume} (\text{make-coroutine}(t, \beta)) \text{ with } x \mapsto u : C \mid \beta : B, \Delta \]

\text{substitution}
\[ \Gamma \vdash [\beta]u[t/x] : \bot \mid \beta : B, \gamma : C, \Delta' \mu \]
\[ \Gamma \vdash \mu x.y.[\beta]u[t/x] : C \mid \beta : B, \Delta' \]

Notice that Crolard’s elimination rule involves an application of the rule *ex falso quodlibet*, which is explicit in the definition of the operator \( \text{resume} \) (see [19], Remark 4.3.). This seems unavoidable working within the \( \lambda\mu \) calculus, but it may not be desirable outside that framework.

Working with the full power of classical logic, if a constructive system of bi-intuitionistic logic is required, then the implication right and subtraction left rules must be restricted; this can be done by considering relevant dependencies. Crolard is able to show that the term assignment for such a restricted logic is a calculus of *safe coroutines*, described as terms in which no coroutine can access the local environment of another coroutine.

Crolard’s work suggests the possibility of defining co-intuitionistic coroutines directly, independently of the typing system of the \( \lambda\mu \)-calculus. Since \( \mu \)-variable abstraction and the \( \mu \)-rule are devices to change the “actual thread” of computation, the effect of removing such rules is that all “threads” of computation are simultaneously represented in a multiple conclusion sequent, but variables \( y \) that are temporarily inaccessible in a term \( N \) are being replaced by a term \( y(M) \) by the substitution \( N[y := y(M)] \), where \( M \) contains a free variable \( x \) which is accessible in the current context. This is the approach pursued in [1, 3, 5] leading to the present categorical presentation.

### 1.2. A dual linear calculus for \( \text{MNJ}^{\setminus \mathcal{P}} \)

We present the grammar and the basic definitions of our dual linear calculus for linear co-intuitionistic logic with subtraction, disjunction and *why not?* operator.

**Definition 1.1.** We are given a countable set of *free variables* (denoted by \( x, y, z \ldots \)), and a countable set of *unary functions* (denoted by \( x, y, z, \ldots \)). The terms of our calculus, denoted by \( R \), are either *m-terms*, denoted by \( M, N \), or *p-terms*, denoted by \( P \).

(i) **Multiplicative** terms, *m-terms* and *p-terms* are defined by the following grammar.
Computational contexts, the basic expressions of our calculus, are contexts satisfying some associative, commutative and has the empty term $\[]$ as the identity. Terms generated by the above grammar the following clauses:

- $M, N := x | x(M) | \text{connect to}(R) | M \circ N | \text{casel}(M) | \text{caser}(M) |$ make-coroutine($M, x) | \].
- $P := \text{postpone}(y \mapsto N, M) | \text{postpone}(M)$.
- $R := M | P$.

(ii) **Multiplicative and exponential terms**, $m$-terms and $p$-terms are obtained by adding to the above grammar the following grammar:

- $M, N := \ldots | [M] | [M, N]$.
- $P := \ldots | \text{store}(P_1, \ldots, P_m, M_1, \ldots, M_n, y_1, \ldots, y_n, x, N)$.

We usually abbreviate “make – coroutine” as “mkc” and “postpone” as “postp”. We often write $\overline{P}$ for a list $P_1, \ldots, P_m$, and similarly $\overline{M}$ for $M_1, \ldots, M_n$. “[]” is the empty term (nil).

**Definition 1.2.** The **free variables** $FV(M)$ in a term $R$ are defined thus:

- $FV(x) = \{x\}$
- $FV(x(M)) = FV(M)$
- $FV(\text{connect to}(R)) = FV(R)$
- $FV(M \circ N) = FV(M) \cup FV(N)$
- $FV(\text{casel}(M)) = FV(\text{caser}(M)) = FV(M)$
- $FV(\text{mkc}(M, x)) = FV(M)$
- $FV(\text{store}(\overline{P}, M_1, \ldots, M_n, y_1, \ldots, y_n, x, N)) = (FV(\overline{P}) \cup (FV(\overline{M})) \setminus \{x\}) \cup FV(N)$
- $FV(\text{casel}(M)) = FV(M) \cup FV(N)$
- $FV(\text{postp}(x \mapsto N, M)) = (FV(N) \setminus \{x\}) \cup FV(M)$.

**Definition 1.3.** Let $\parallel$ be a binary operation on terms (parallel composition) which is associative, commutative and has the empty term $\[]$ as the identity. Terms generated by (zero or more) applications of parallel composition are called **contexts**. Thus **contexts** are generated by the following grammar:

- $C := R \mid (C \parallel R)$

modulo the structural congruences

- (i) $R_0 \parallel (R_1 \parallel R_2) \equiv (R_0 \parallel R_1) \parallel R_2$.
- (ii) $R_0 \parallel R_1 \equiv R_1 \parallel R_0$.
- (iii) $(R_0 \parallel []) \equiv R_0$.
- (iv) $C_0 \parallel R \parallel C_1 \equiv C_0 \parallel R' \parallel C_1$ if $R \equiv R'$.

Let $R_1 \parallel \ldots \parallel R_k$ be a context, where all $R_i$ are non-null, $i \leq k$. Notice that the notation is well-defined by generalized associativity. We write $\overline{R} : R_1 \parallel \ldots \parallel R_k$ if all free variables occurring in $R_1, \ldots, R_k$ are in the list $\overline{R}$.

**Computational contexts**, the basic expressions of our calculus, are contexts satisfying some correctness conditions, that guarantee the identification of a context and rule out circular structures. Our “calculus of coroutines” is used here in a typed setting, where self referential structures are not needed.

**Definition 1.4.** An expression $S_x : R_1 \parallel \ldots \parallel R_k$ is a (correct) **computational context** if it satisfies the following axioms.

- (1) Each term in the set $\{R_1, \ldots, R_k\}$ contains $x$ and no other free variable.
(2) In every term of the form \( \text{postp}(y \mapsto N, M) \) the term \( N \) contains a free variable \( y \) with \( y \notin \text{FV}(M) \) and no other free variable.

(3) In every term \( \text{store}(T, N_1, \ldots, N_n, y_1, \ldots, y_n, z, M) \) the terms \( N_i \) are of the forms \([N]\) or \( \text{connect to}(R) \) for some \( N \) or \( R \).

(4) Let \( \mathcal{S} = \text{store}(T, N_1, \ldots, N_n, y_1, \ldots, y_n, z, M) \) occur within a multiplicative computational context \( \mathcal{S}_x \); write \( \mathcal{S}_x^- \) for \( \mathcal{S}_x \) without \( \mathcal{S} \). Then \( \{T, N_1, \ldots, N_n\} \) is a computational context \( \mathcal{S} \) for some free variable \( z \) with \( z \neq x \). We say that \( \mathcal{S} \) occurs immediately within \( \mathcal{S} \).

(5) In a computational context \( \mathcal{S}_x \) the nesting of \( p \)-terms of the form \( \text{store} \) within \( \mathcal{S}_x \) has the structure of a rooted tree, with root \( \mathcal{S}_x^- \) itself.

A computational context is said multiplicative if it does not contain \( \text{store} \) terms.

**Remark 1.5.** By axiom 1, the relevant components of a computational context are uniquely identified. Axioms 2 is analogue to the acyclicity condition in proof nets for linear logic. Axioms 3 and 4 induce a structure on context that corresponds to that of boxes in proof nets. Axiom 5 characterizes exponential boxes in our framework.

**Definition 1.6** (\( \alpha \)-equivalence). Let \( \mathcal{S}_x \) and \( \mathcal{S}_x' \) be computational contexts. To define what it means that \( \mathcal{S}_x \) and \( \mathcal{S}_x' \) are \( \alpha \)-equivalent, we need to define this property for sets of terms \( \mathcal{S}_T \) and \( \mathcal{S}_T' \) that may contain more than one free variable from the lists \( T \) and \( T' \), respectively; therefore they are not correct computational contexts. The definition is by induction on the number of terms occurring in \( \mathcal{S}_T \).

1. If \( \mathcal{S}_x = \{x\} \), then \( \mathcal{S}_x \equiv \mathcal{S}_x' \) iff \( \mathcal{S}_x' = \{x'\} \) and \( x = x' \).
2. If \( \mathcal{S}_T = \{x(M)\} \), then \( \mathcal{S}_T \equiv \mathcal{S}_T' \) iff \( \mathcal{S}_T' = \{x(M')\} \) and \( M \equiv M' \). A similar definition applies if \( \mathcal{S}_x = \{\text{connect to}(M)\} \) or \( \{\text{case1}(M)\} \) or \( \{\text{case2}(M)\} \) or \( \{[M]\} \) or \( \{\text{postp}(P)\} \).
3. If \( \mathcal{S}_T = \{M \varphi N\} \), then \( \mathcal{S}_T \equiv \mathcal{S}_T' \) iff \( \mathcal{S}_T' = \{M' \varphi N'\} \) and \( M \equiv M' \) and \( N \equiv N' \). A similar definition applies if \( \mathcal{S}_x = \{[M, N]\} \).
4. Let \( \mathcal{S}_T \) be partitioned as
   \[
   \mathcal{S}_T = \mathcal{S}_T^- \cup \{\text{mkc}(M, y)\} \cup \mathcal{S}_{Ty}[y := y(M)];
   \]
   then \( \mathcal{S}_T \equiv \mathcal{S}_T' \) iff \( \mathcal{S}_T' \) can be partitioned as \( \mathcal{S}_T' = \mathcal{S}_T'^- \cup \{\text{mkc}(M', y)\} \cup \mathcal{S}_{Ty'}[y' := y(M')] \) and \( \mathcal{S}_T^- \cup \{M\} \equiv \mathcal{S}_T'^- \cup \{M'\} \) and, moreover, for all variables \( v \) except for a finite number \( \mathcal{S}_{Ty}[y := v] \equiv \mathcal{S}_{Ty'}[y' := v] \).
5. Let \( \mathcal{S}_T \) be partitioned as
   \[
   \mathcal{S}_T = \mathcal{S}_T^- \cup \{\text{postpone}(y \mapsto N, M)\} \cup \mathcal{S}_{Ty}[y := y(M)];
   \]
   then \( \mathcal{S}_T \equiv \mathcal{S}_T' \) iff \( \mathcal{S}_T' \) can be partitioned as \( \mathcal{S}_T' = \mathcal{S}_T'^- \cup \{\text{postpone}(y' \mapsto N', M')\} \cup \mathcal{S}_{Ty'}[y' := y'(M')] \) and \( \mathcal{S}_T^- \equiv \mathcal{S}_T'^- \) and, moreover, for all variables \( v \) except for a finite number \( \mathcal{S}_{Ty}[y := v] \equiv \mathcal{S}_{Ty'}[y' := v] \).
6. Let \( \mathcal{S}_T \) be partitioned as \( \mathcal{S}_T = \mathcal{S}_T^- \cup \{S_1, \ldots, S_k\} \) where for \( i \leq k \), \( S_i \) is a \( \text{store} \) term with a set of terms \( \mathcal{S}_{Ti} \) immediately inside it. Then \( \mathcal{S}_T \equiv \mathcal{S}_T' \) iff \( \mathcal{S}_T' \) can be partitioned in a similar way as \( \mathcal{S}_T' = \mathcal{S}_T'^- \cup \{S'_1, \ldots, S'_k\} \) where for \( i \leq k \) the set \( \mathcal{S}_{Ti} \) occurs immediately inside \( S'_i \), for \( i \leq k \) and, moreover
   \[
   (i) \quad \mathcal{S}_{Ti} \equiv \mathcal{S}_{Ti}'; \n   (ii) \quad \text{for all } i \leq k \text{ and for all variables } v \text{ except for a finite number } \mathcal{S}_{Ti}[z_i := v] \equiv \mathcal{S}_{Ti'}[z_i' := v].
   \]
Remarks 1.7. (i) Consider the terms make – coroutine, binary postpone and store. They are binders acting on a whole computational context, rather than on a delimited scope within a single term. We may call their action remote binding; it is expressed by a substitution of some \( m \)-term \( x(M) \) for the free variable \( x \) throughout a computational context \( S_x \). One could express remote binding by a more familiar notation, as in the \( \lambda \)-calculus or in the \( \pi \)-calculus; but then the scope of a binder would partition a context and parallel composition would not appear as a top-level operator only. In the typed case one could not assign a separate \( m \)-term to each formula in the succedent: at best, one could assign an "access port" to a unique term assigned to the whole sequent, as in the translations of linear logic into the \( \pi \)-calculus (see [10]). This would be against a main motivation of this calculus, namely, to give a "distributed term assignment" for a "multiple conclusion" co-intuitionistic deductive system. An application of our notation for "remote binding" is found in section 2 on a probabilistic interpretation of subtraction and par.

(ii) The \( p \)-terms binary postpone and store are also local binders of the free variable occurring in their argument. Indeed in a term \( \text{postpone}(y \mapsto N, M) \) a free variable \( y \) occurring in \( N \) becomes locally bound; then \( y \) is replaced by \( y(M) \) in the computational context, to express remote binding. In a term \( \text{store}(\bar{P}, N_1, \ldots, N_n, y_1, \ldots, y_n, z, M) \) a free variable \( z \) becomes bound; then a term \( z(N_i) \) occurs in the term \( y_i(z(N_i)) \), that replaces the stored \( N_i \) in the computational context. A more complete notation of our \( p \)-terms would be \( \text{postpone}(y \mapsto N, M) \) with \( y \) for \( y \) and \( \text{store}(\bar{P}, N_1, \ldots, N_n, y_1, \ldots, y_n, z, M) \) with \( z \) for \( z \), explicitly establishing the connection between the locally bound variable and its corresponding unary function. For terms of the form store we are spared the more verbose notation by the acyclicity axioms, by which the locally bound variable is uniquely identified.

(iii) In the notation of \( p \)-terms of the form \( \text{postpone}(y \mapsto N, M) \) it is sometimes convenient to ignore the distinction between local binding and remote binding and treat the occurrences in \( N \) of the variable \( y \) as remotely bound; namely, we can write \( \text{postpone}(y \mapsto N[y := y(M)], M) \). Indeed the distinction between local and remote binding can be recovered from the structure of the terms. This notation allows us to adopt a tree-like notation for decorated co-intuitionistic Natural Deduction derivations analogue to that of Prawitz's Natural Deduction trees decorated with \( \lambda \)-terms. On the other hand, such a treatment seems unmanageable for \( p \)-terms of the form store; for them we retain a box-like notation.

Definition 1.8. Substitution of a term \( t \) for a free variable \( x \) in a term \( R \) is defined as follows:

\[
\begin{align*}
x[x := M] & = M, & y[x := M] & = y \text{ if } x \neq y; \\
\text{connect to}(R)[x := M] & = \text{connect to}(R[x := M]) \\
\text{postp}(N)[x := M] & = \text{postp}(N[x := M]) \\
y(N)[x := M] & = y(N[x := M]); \\
(N_0 \varphi N_1)[x := M] & = (N_0[x := M]) \varphi (N_1[x := M]) \\
\text{casel}(N)[x := M] & = \text{casel}(N[x := M]), \\
\text{caser}(N)[x := M] & = \text{caser}(N[x := M]); \\
\text{mkc}(N,y)[x := M] & = \text{mkc}(N[x := M], y), \\
\text{store}(\bar{N}, \bar{y}, z, N)[x := M] & = \text{store}(\bar{N}[x := M], \bar{y}, z, N[x := M]) \\
\text{postp}(y \mapsto (N_1), N_0)[x := M] & = \text{postp}(y \mapsto (N_1[x := M]), N_0[x := M]). \\
[R][x := M] & = [R[x := M]] \\
[R_0, R_1][x := M] & = [R_0[x := M], R_1[x := M]]
\end{align*}
\]

Proposition 1.9.
The operation of $\beta$-terms satisfies all the axioms in definition 1.4. In all cases the proposition is easily proved by checking that the resulting set of

Proof. In all cases the proposition is easily proved by checking that the resulting set of terms satisfies all the axioms in definition 1.4. □

The operation of $\beta$-reduction transforms a computational context $S_x$ into a computational context $S'_x$. It may be either local, affecting only the terms where the redex occurs, or a global operation with side-effects on parts of $S_x$, mainly relabelling the terms that express binding by make – coroutine, postpone or store.

Definition 1.10. $\beta$-reduction of a redex $Red$ in a computational context $S_x$ is defined as follows.

(i) If $Red$ is a $m$-term $N$ of the following form, then the reduction is local and consists of the rewriting $N \rightsquigarrow_{\beta} N'$ in $S_x$ as follows:

\[
\text{postp}(\text{connect to}(R)) \rightsquigarrow_{\beta} [],
\]
\[
\text{casel} (N_0 \varphi N_1) \rightsquigarrow_{\beta} N_0; \quad \text{caser} (N_0 \varphi N_1) \rightsquigarrow_{\beta} N_1.
\]
If the principal operator of \( \text{Red} \) is a binary \textit{postpone} or \textit{store}, then the reduction is global and consists of the following rewriting. By the axioms in definition 1.3, \( \text{Red} \) occurs inside a computational context \( S_v \) in the rooted tree of nested \( p \)-terms of \( S_x \) and the rewriting takes place within \( S_v \).

(ii) If \( \text{Red} \) has the form \( \text{postp}(z \mapsto N, \text{mkc}(M, y)) \), then \( S_v \) is partitioned as

\[
S_v = \text{Red} \cup S_{vyz}[y := y(M), z := \text{mkc}(M, y)]
\]

(a simultaneous substitution of \( y(M) \) for \( y \) and of \( \text{mkc}(M, y) \) for \( z \) in \( S_{vyz} \)). Then a \textit{reduction} of \( \text{Red} \) transforms the computational context as follows:

\[
S_v = S_{vyz}[y := N[z := M], z := M].
\]

(iii) If \( \text{Red} \) is a term with principal operator \textit{store}, then \( S_v \) is partitioned as \( S_v^- \cup S \) where

\[
S = \text{store}(\overline{P}, N_1, \ldots, N_n, y_1, \ldots, y_n, z, N)
\]

Here \( N \) is either \([M]\) or \textit{connect to}(\( R \)) or \([[[M_0][M_1]]]\).

By the axioms 1.4, \( \overline{P}, N_1, \ldots, N_n \) form a computational context \( S_z \). Moreover \( y_1(z([M])), \ldots, y_n(z([M])) \) may occur in \( S_v^- \) so we write \( S_v^- \) as \( S_{v'y_1 \ldots y_nv} \[y_1 := y_1(z(M)), \ldots, y_n := y_n(z(M))\] \).

- If \( N = [M] \), then
  \[
  S_v \sim_\beta S_{v'y_1 \ldots y_nv} \[y_1 := N_1[z := M], \ldots, y_n := N_n[z := M]\] \cup \{(\overline{P}[z := M]\).
  \]

- If \( N = \text{connect to}(R) \), where \( R \) belongs to \( S_v^- \), then
  \[
  S_v \sim_\beta S_{v'y_1 \ldots y_nv} \[y_1 := \text{connect to}(R), \ldots, y_n := \text{connect to}(R)\]
  \]

- If \( N = [M_0, M_1] \), then
  \[
  S_v \sim_\beta S_{v'y_1 \ldots y_nv} \[y_1 := [y_1(z(M_0))], [y_1(z(M_1))], \ldots, y_n := [y_n(z(M_0))], [y_n(z(M_1))]\]
  \[
  \cup \{\text{store}(\overline{N, y, z, M_0}), \text{store}(\overline{N, y, z, M_1})\}
  \]

\textbf{Remarks 1.11.} Here are some informal explanations about our calculus and notations.

(i) In our “distributed” model of computation a redex arises when an \( m \)-term \( M \) becomes part of another term; the rewriting of the redex has global effect. On the contrary, a \( p \)-term can only be sub-term of a \( p \)-term of the form \textit{store}, but such nesting has only a structural significance; no redex is created in this way. A \( p \)-term \( P \) sits in the “control area”, waiting to become active as a redex if a suitable \( m \)-term is substituted inside it. A term \( y(M) \) denotes a variable \( y \) that has become bound because of an operation in which the term \( M \) is active; in some sense \( y(M) \) is an input which is no longer accessible. Later in the computation such an input may become active again in a term \( R \) and ready for a substitution by a \( m \)-term \( N \) in a rewriting of the form \( R[y := y(M)] \sim R[y := N] \).

(ii) When a \( p \)-term \( \text{store}(\overline{P}, N_1, \ldots, N_n, y_1, \ldots, y_n, z, N) \) is created, the \( m \)-terms \( N_1, \ldots, N_n \) are set aside, together with the local \( p \)-terms \( P_1, \ldots, P_m \), within the new \textit{store} terms sitting in the control area, but the “guarding terms” \( y_1, \ldots, y_n \) associated with \( N_1, \ldots, N_n \) remain active, since they are part of other terms in the context. Also the free variable \( z \) occurring in the terms \( P_j \) and \( N_i \) becomes inaccessible and is substituted with \( z(N) \). Only the term \( N \) is active in the storage operation. If \( N = [M] \) then the computation is reactivated in
The term assignment to co-ILL in sequent-style Natural Deduction notation is given in tables 2 and 3. Sequents are of the form

\[ x : E \vdash P \mid M : \Gamma \]

where

- the area of the succedent to the left of “\|” may be called “control area”;
- \( \overline{P} = P_1, \ldots, P_m \) is a sequence of \( p \)-terms;
- \( \overline{M} : \Gamma \) stands for \( M_1 : C_1, \ldots, M_n : C_n \), where \( \Gamma = C_1, \ldots, C_n \);
- if \( \overline{R} = R_1, \ldots, R_n \) then \( \overline{R}[x := N] \) stands for \( R_1[x := N], \ldots, R_n[x := N] \);
- We shall also use the abbreviation \( \kappa : \Gamma \) for \( \overline{P} \mid \overline{M} : \Gamma \). If also \( \zeta : \Delta \) stands for \( \overline{Q} \mid \overline{N} : \Delta \), then \( \kappa : \Gamma, \zeta : \Delta \) stands for \( \overline{P}, \overline{Q} \mid \overline{M} : \Gamma, \overline{N} : \Delta \).

1.3. Examples of multiplicative contexts.

1. The following computational context \( S_x \)

\[ S_x = \text{postp}(x) \parallel \text{connect to}(\text{postp}(x)) \]

is correct. It is typed as follows:

\[
\begin{align*}
\bot \text{-elim} \\
\quad \quad \frac{x \vdash \bot \quad \text{postp}(x)}{x : \bot \vdash \text{postp}(x) \mid \text{connect to}(\text{postp}(x)) : \bot} & \quad \bot \text{-intro}
\end{align*}
\]

This derivation may be regarded as the \( \eta \)-expansion of the axiom

\[ x : \bot \vdash x : \bot. \]

2. Given the computational contexts \( S_x = x \parallel \text{connect to}(x) \) and \( S_y = \text{postp}(y) \), we obtain a correct computational context by substitution of \( \text{connect to}(x) \) for \( y \) in \( S_y \):

\[ S'_x = x \parallel \text{postp}(\text{connect to}(x)) \]

\( S'_x \) \( \beta \)-reduces to \( S''_x = x \parallel [\cdot] \).

3. The following context \( S'_z \) is not correct: it violates Axiom 2 in definition

\[ S'_z = \text{postp}(y \mapsto [x(y), x(z)], \text{mkc}(z, x)) \parallel \text{mkc}(y(t), x) \]
Here we write $t$ for $\text{mkc}(z,x)$. The following context $S_z$ is correct

$$S_z = \text{postp}(y \mapsto x(y), \text{mkc}(z,x)) \parallel \text{mkc}(y(t),x) \parallel x(z)$$

and is typed as follows:

$$\frac{z : C \vdash z : C \quad x : C \vdash x : C}{z : C \vdash \text{mkc}(z,x) : C \& C, x(z) : C} \quad \perp \hspace{-0.6cm} \text{I}$$

$$\frac{y : C \vdash y : C \quad x : C \vdash x : C}{y : C \vdash \text{mkc}(y,x) : C \& C, x(y) : C} \quad \perp \hspace{-0.6cm} \text{I}$$

$$\frac{z : C \vdash \text{postp}(y \mapsto x(y), \text{mkc}(z,x)) \mid \text{mkc}(y(t),x) : C \& C, x(z) : C}{z : C \vdash x(z) : C} \quad \perp \hspace{-0.6cm} \text{E}$$

One could check that the above derivation is dual to the derivation

$$f : A \to A, x : A \vdash (\lambda x.fx)x : A$$
Table 3: Decorated Natural Deduction for co-ILL exponential

in the simply typed λ-calculus. A more substantial example of computation in our typed dual linear calculus is given in Appendix, Section A.


In our setting co-intuitionistic logic admits a simple probabilistic interpretation which fits well in the view of co-intuitionism as a logic of hypotheses. Indeed if co-intuitionistic logic is about the justification properties of hypotheses, then the co-intuitionistic consequence relation must be about the preservation of probability assignments from the premise to the conclusions; a term calculus for such a logic must allow us to compute probabilities and verify the preservation property. We sketch our result only for the multiplicative linear fragment, i.e., for typing derivations in the linear system with subtraction and par only.

We find it easier to state our result for a decorated sequent calculus for multiplicative co-intuitionistic linear logic. Such a calculus is equivalent to our system of decorated sequent-style natural deduction; in fact its right rules coincide with the introduction rules and using cut the left rules given below are shown to be equivalent to the elimination rules.

Definition 2.1. To the judgements of linear co-intuitionistic logic we assign events in a probabilistic setting. We write $\overline{C}$, $C \cap D$ and $C \cup D$ for complementation, intersection and union between events; there is an impossible event $\emptyset$ and a certain event $\overline{\emptyset}$. A probabilistic assignment is a map $(\ )^P : \text{judg} \to \text{events}$ satisfying the following constraints:

- if $(M : C)^P = C$ and $(x : D)^P = D$, then $\text{mkc}(M, x) : C ÷ D)^P = C \cap \overline{D}$;
- if $(M_0 : C_0)^P = C_0$ and $(M_1 : C_1)^P = C_1$, then $(M_0 \triangleleft M_1 : C_0 \triangleleft C_1)^P = C_0 \cup C_1$;
- if $(M : C_0 \triangleleft C_1)^P = C$ then $(\text{casel}(M) : C_0)^P \subseteq C$ and $(\text{caser}(M) : C_1)^P \subseteq C$;
- $(\text{postp}(M))^P = \emptyset$ and $(\text{postp}(x \mapsto M, N))^P = \emptyset.$
Proposition 2.2. (Decomposition property) Let $d$ be a sequent calculus derivation of $x : H \vdash t_1 : C_1, \ldots, t_n : C_n$ and let $(\mathcal{P}) : \text{judge} \rightarrow \text{events}$ be an assignment to the judgements in $d$ satisfying the constraints of Definition 2.1, and suppose $H, C_1, \ldots, C_n$ are assigned to $x : H \vdash t_1 : C_1, \ldots, t_n : C_n$. There are pairwise disjoint events $C'_1 \subseteq C_1, \ldots, C'_n \subseteq C_n$ such that

$$(C'_1 \cup \cdots \cup C'_n) \cap H = H.$$ \[\text{The events } C'_1, \ldots, C'_n \text{ can be constructed from the dependencies of the terms } t_1, \ldots, t_n.\]

Example 2.3. Consider the following very simple example:

$$\begin{array}{c}
x : C \vdash x : C & y : D \vdash y : D \\
x : C \lor \text{mkc}(x, y) : C \land D, y(x) : D
\end{array}$$

If the event $C$ is assigned to $x : C$, and $D$ is assigned to $y : D$, but also to $y(x) : D$, then we have the following inclusions

$$\begin{array}{c}
C \subseteq C & D \subseteq D \\
C \subseteq (C \land D) \cup D
\end{array}$$

We have equality only by assigning $C \land D$ to $y(x) : D$, as suggested by the dependency of $y(x) : D$ on $x : C$.

$$\begin{array}{c}
C = C & D = D \\
C = (C \land D) \cup (C \land D)
\end{array}$$

Proof. By induction on $d$. The case of assumptions $x : H \vdash x : H$ is obvious and that of cut is immediate from the inductive hypothesis.

Subtraction right: by inductive hypothesis we may assume that the assignments to the premises $v : E \lor \kappa : \Gamma, M : C$ and $x : D \lor \zeta : \Delta$ satisfy the conditions of the lemma, i.e., that $((\bigcup \Gamma) \cup C) \cap E = E$ and $(\bigcup \Delta) \cap D = D$, where the events in $\Gamma$ are pairwise disjoint and so are those in $\Delta$. In the term assignment to the conclusion

$$v : E \lor \kappa : \Gamma, \text{mkc}(M, x) : C \land D, \zeta[x := x(M)] : \Delta$$

in all terms $\zeta : \Delta$ the variable $x : D$ has been replaced by $x(M)$, where $M : C$. We interpret this fact as the instruction that in the context of the conclusion the disjoint events $D'_j \subseteq D_j$ must be $D_j \cap (C \land D)$ for each $D_j \in \Delta$. Then

$$C = (C \land D) \cup (C \land D) = (C \land D) \cup (C \land D \land \bigcup \Delta) = (C \land D) \cup (\bigcup D'_j)$$

hence $C \land E = [(C \land D) \land E] \cup [(\bigcup D'_j) \land E]$. Thus

$$((\bigcup D'_j) \cup (C \land D) \cup (\bigcup D'_j)) \cap E = E.$$
In the case of par right there is nothing to prove; in the case of par left we only need to make sure that the events \( C_0 \) and \( C_1 \) assigned to \( C_0 \) and \( C_1 \) are disjoint. If not, assign \( C_0 \) to \( C_0 \) and \( C_1 \cap C_0 \) to \( C_1 \), or alternatively \( C_0 \cap C_1 \) to \( C_0 \) and \( C_1 \) to \( C_1 \).

Remarks 2.4. (i) Since events are assigned to expressions \( t : X \) rather than to formulas \( X \), if \( t : X \) and \( u : X \) occur in the same context then \((t : X)^P\) and \((u : X)^P\) are events that may or may not be disjoint of each other.

(ii) The common sense reading of the co-intuitionistic consequence relation \( H \vdash C_1, \ldots, C_n \) is as follows.

If it is justified to make the hypothesis \( H \), then it is justified to make the hypotheses \( C_1, \ldots, C_n \).

The probabilistic interpretation gives a mathematical counterpart of this reading.

If the probability of the event \( H \) assigned to \( x : H \) is greater than zero, then the conditional probability of the union of the events \( C_1, \ldots, C_n \) assigned to \( t_1 : C_1, \ldots, t_n : C_n \), given \( H \) is equal to one.

The indexing of the terms \( t_1, \ldots, t_n \) can be regarded as computational devices for verifying such an interpretation in the sense of the Decomposition Property.

3. Categorical Semantics

We recall the definition of a symmetric monoidal category.

Definition 3.1. A symmetric monoidal category (SMC) \((\mathbb{C}, \bullet, 1, \alpha, \rho, \gamma)\), is a category \( \mathbb{C} \) equipped with a bifunctor \( \bullet : \mathbb{C} \times \mathbb{C} \to \mathbb{C} \) with a neutral element \( 1 \) and natural isomorphisms \( \alpha, \rho \) and \( \gamma \):

1. \( \alpha_{A,B,C} : A \bullet (B \bullet C) \sim (A \bullet B) \bullet C \);
2. \( \lambda_A : 1 \bullet A \sim A \)
3. \( \rho_A : A \bullet 1 \sim A \)
4. \( \gamma_{A,B} : A \bullet B \sim B \bullet A \).

which satisfy the following coherence diagrams.

\[
\begin{array}{ccc}
A \bullet (B \bullet (C \bullet D)) & \xrightarrow{\alpha_{A,B,C} \bullet D} & (A \bullet B) \bullet (C \bullet D) \\
\downarrow \alpha_{A,B,C} & & \downarrow \alpha_{A,B,C} \\
A \bullet ((B \bullet C) \bullet D) & \xrightarrow{\alpha_{A,B,C} \bullet D} & (A \bullet (B \bullet C)) \bullet D
\end{array}
\]
The following equality is also required to hold: $\lambda_1 = \rho_1 : 1 \cdot 1 \rightarrow 1$.

Given a signature $Sg$, consisting of a collection of types $\sigma_i$ and a collection of sorted function symbols $f_j : \sigma_1, \ldots, \sigma_n \rightarrow \tau$ and given a SMC $(\mathcal{C}, \cdot, 1, \alpha, \lambda, \rho, \gamma)$, a structure $\mathcal{M}$ for $Sg$ is an assignment of an object $[\sigma]$ of $\mathcal{C}$ for each type $\sigma$ and of a morphism $[[f]] : [[\sigma_1]] \cdot \ldots \cdot [[\sigma_n]] \rightarrow [[\tau]]$ for each function $f : \sigma_1, \ldots, \sigma_n \rightarrow \tau$ of $Sg$.

The types of terms in context $\Delta = [M_1 : \sigma_1, \ldots, M_n : \sigma_n]$ are interpreted as $[[\sigma_1, \sigma_2, \ldots, \sigma_n]] = (\ldots ([[\sigma_1]] \cdot [[\sigma_2]]) \ldots) \cdot [[\sigma_n]]$; left associativity is also intended for concatenations of type sequences $\Gamma, \Delta$. Thus we need the following functions $\text{Split}(\Gamma, \Delta) : [[\Gamma]] \cdot [[\Delta]] \\ \rightarrow [[\Gamma \cdot \Delta]]$

$$\text{Split}(\Gamma, \Delta) \begin{cases} \lambda_\Delta^{-1} & \text{if } \Gamma = \emptyset \\ \rho_\Gamma & \text{if } \Delta = \emptyset \\ \text{id}_\Gamma \cdot A & \text{if } \Delta = A \\ \text{Split}(\Gamma, \Delta') \cdot \text{id}_A ; \alpha_{\Gamma', \Delta, A}^{-1} & \text{if } \Delta = \Delta', A. \end{cases}$$

and $\text{Join}(\Gamma, \Delta) : [[\Gamma]] \cdot [[\Delta]] \rightarrow [[\Gamma, \Delta]]$

$$\text{Join}(\Gamma, \Delta) \begin{cases} \lambda_\Delta & \text{if } \Gamma = \emptyset \\ \rho_\Gamma & \text{if } \Delta = \emptyset \\ \text{id}_{\Gamma \cdot A} & \text{if } \Delta = A \\ \alpha_{\Gamma, \Delta', A} ; \text{Join}(\Gamma, \Delta') \cdot \text{id}_A & \text{if } \Delta = \Delta', A. \end{cases}$$

Similarly we have $\text{Split}_n(\Gamma_1, \ldots, \Gamma_n) : [[\Gamma_1]] \cdot \ldots \cdot [[\Gamma_n]]$ and $\text{Join}_n$.

The semantics of terms in context is then specified by induction on terms:

$$[x : \sigma \triangleright x : \sigma] = df \text{id}[\sigma]$$

$$[x : \sigma \triangleright f(M_1, \ldots, M_n) : \tau] = df \left[ x : \sigma \triangleright M_1 : \sigma_1 \right] \cdot \ldots \cdot \left[ x : \sigma \triangleright M_n : \sigma_n \right]; [f]$$

The Exchange right rule is handled implicitly by symmetry in the model (see [13], Lemma 13):

$$[x : \sigma \triangleright \overline{M} : \Gamma, N : \tau, M : \sigma] = [[x : \sigma \triangleright \overline{M} : \Gamma, M : \sigma, N : \tau]]; \alpha_{\Gamma, \sigma, \tau}^{-1} \cdot \text{id}_M \cdot \gamma_{\sigma, \tau}; \alpha_{\Gamma, \tau, \sigma}$$
(Notice that, as a notational convenience, we are sometimes reusing the names of types to denote their interpretation as objects). One then proves by induction on the type derivation that substitution in the term calculus corresponds to composition in the category ([13], Lemma 13):

**Lemma 3.2.** Let \( x : \sigma \triangleright M : \Gamma, M : \tau \) and \( y : \tau \triangleright N : \Delta \) be derivable terms in context, then
\[
[x : \sigma \triangleright M : \Gamma, N[y := M] \Delta] = [x : \sigma \triangleright M : \Gamma, M : \tau]; \text{id}_{\Gamma} \bullet [y : \tau \triangleright N : \Delta]; \text{Join}(\Gamma, \Delta)
\]

Let \( M \) be a structure for a signature \( Sg \) in a SMC \( C \). Given an equation in context for \( Sg \)
\[
x : \sigma \triangleright M : \Gamma, M = N : \tau
\]
we say that the structure satisfies the equation if the morphisms assigned to \( x : \sigma \triangleright M : \Gamma, M : \tau \) and to \( x : \sigma \triangleright M : \Gamma, N : \tau \) are equal. Then given an algebraic theory \( Th = (Sg, Ax) \), a structure \( M \) for \( Sg \) is a model for \( Th \) if it satisfies all the axioms in \( Ax \).

**Lemma 3.3.** Let \( C \) be a SMC, \( Th \) an algebraic theory and \( M \) a model of \( Th \) in \( C \). Then \( M \) satisfies the equations in context in Table 4.

\[
\begin{align*}
z : D \triangleright M : \Gamma, M : \sigma & \quad \text{Refl} \quad z : D \triangleright M : \Gamma, M = N : \sigma \quad \text{Symm} \\
z : D \triangleright M : \Gamma, M = M : \sigma & \\
z : D \triangleright M : \Gamma, M_0 = M_1 : \sigma & \quad z : D \triangleright M : \Gamma, M_1 = M_2 : \sigma \quad \text{Trans} \\
z : D \triangleright M : \Gamma, M_0 = M_1 : \sigma & \\
z : D \triangleright M : \Gamma, N[x := M_0] = N[x := M_1] : \Delta, N_0[x := M_0] = N_1[x := M_1] : \tau \quad \text{Subst}
\end{align*}
\]

Table 4:

3.1. **Analysis of the rules of co-intuitionistic linear logic.** We work with symmetric monoidal categories satisfying the dual condition to closure, namely, with monoidal categories of the form \((C, \bullet, \cdot, 1, \alpha, \lambda, \rho, \gamma)\) such that for all objects \( A \) in \( C \), the functor \( A \cdot − \) has a left adjoint \(- \cdot A \). We call such monoidal categories left closed.

Given a symmetric monoidal category \( C \), its opposite is also symmetric monoidal. If \( C \) is closed, i.e., \( A \cdot − \) has a right adjoint, then certainly \( C^{op} \) has a left adjoint. It is well-known that in a symmetric monoidal closed category \( C \) we can construct a model of multiplicative intuitionistic linear logic, hence it is certainly not surprising that a model of multiplicative co-intuitionistic linear logic may be constructed in \( C^{op} \). The point of the exercise that follows, however, is to check that the dual linear calculus given above in Section 1.2 is indeed suitable for the construction of such an interpretation. We consider the rules for each connective in turn.
3.2. Linear disjunction Par. \[3.2.1\]

**Par introduction.** The introduction rule for Par is of the form

\[
\frac{x : D \triangleright \kappa : \Theta, M_0 : A, M_1 : B}{x : D \triangleright \kappa : \Theta, M_0 \circ M_1 : A \circ B}
\]

This suggests an operation on Hom-sets of the form

\[
\Phi_{D, \Theta} : \mathbb{C}(D, \Theta \triangleleft A \triangleleft B) \rightarrow \mathbb{C}(D, \Theta \triangleleft A \circ B)
\]

natural in \(\Theta\) and \(D\). Given \(e : D \rightarrow \Theta \triangleleft A \triangleleft B\), \(d : D' \rightarrow D\) and \(h : \Theta \rightarrow \Theta'\), naturality yields

\[
\Phi_{D', \Theta'}(d; e; h \circ id_A \circ id_B) = d; \Phi_{D, \Theta}(e) ; h \circ id_{A \circ B}
\]

In particular, letting \(e = id_{\Theta} \circ id_A \circ id_B\), \(d : D \rightarrow \Theta \triangleleft A \triangleleft B\) and \(h = id_{\Theta}\) we have

\[
\Phi_{D, \Theta}(d) = d; \Phi_{\Theta}(id_A \circ id_A \circ id_B)
\]

By functoriality of \(\triangleleft\) we have \(id_A \circ id_B = id_{A \triangleleft B}\). Hence, writing Par for \(\Phi_{\Theta}(id_A \circ id_A \circ B)\) we have \(\Phi_{D, \Theta}(d) = d\); Par. We define

\[
[x : D \triangleright \kappa : \Theta, M_0 : A, N : B] =_{df} [x : D \triangleright \kappa : \Theta, M : A, N : B]; \text{Par}.
\]

**Par elimination.** The Par elimination rule has the form

\[
\frac{z : D \triangleright \kappa : \gamma, \Lambda : A \circ B}{x : A \triangleright \zeta : \Gamma \quad y : B \triangleright \xi : \Delta} \quad \phi \ E
\]

This suggests an operation on Hom-sets of the form

\[
\Psi_{D, \gamma, \Gamma, \Delta} : \mathbb{C}(D, \gamma \triangleleft \Lambda \triangleleft \Delta) \times \mathbb{C}(A, \Gamma) \times \mathbb{C}(B, \Delta) \rightarrow \mathbb{C}(D, \gamma \triangleleft \Lambda \triangleleft \Delta)
\]

natural in \(D, \gamma, \Gamma, \Delta\). Given morphisms \(g : D \rightarrow \gamma \triangleleft \Lambda \triangleleft \Delta\), \(e : A \rightarrow \Gamma\) and \(f : B \rightarrow \Delta\) and also \(a : D' \rightarrow D\), \(p : \gamma \rightarrow \gamma'\), \(c : \Gamma \rightarrow \Gamma'\) and \(d : \Delta \rightarrow \Delta'\) naturality yields

\[
\Psi_{D', \gamma', \Gamma', \Delta'}((a ; g ; p \circ id_{A \circ B}), (e ; c), (f ; d)) =
\]

\[
a ; \Psi_{D, \gamma, \Gamma, \Delta}(g, e, f) ; p \circ c \circ d ; \text{Join}(\gamma', \Gamma', \Delta').
\]

In particular, setting \(e = id_A\), \(f = id_B\) and also \(a = id_D\), \(p = id_{\gamma}\), we get

\[
\Psi_{D, \gamma, \Gamma, \Delta}(g, c, d) = \Psi_{D, \gamma, \Gamma, \Delta}(g, id_A, id_B); id_{\gamma} \circ c \circ d ; \text{Join}(\gamma, \Gamma, \Delta)
\]

Writing \((g)^*\) for \(\Psi_{D, \gamma, \Gamma, \Delta}(g, id_A, id_B)\) we define

\[
[z : D \triangleright \kappa : \gamma, \Lambda [x := \text{casel } N] : \Gamma, \xi [y := \text{caser } N] : \Delta] =_{df}
\]

\[
[z : D \triangleright \kappa : \gamma, \Lambda : A \circ B]^*; id_{\gamma} \circ [x : A \triangleright \zeta : \Gamma] \circ [y : B \triangleright \xi : \Delta]; \text{Join}(\gamma, \Gamma, \Delta).
\]

**3.2.3. Equations in context.** We have equations in context of the form

<table>
<thead>
<tr>
<th>(\varphi - \beta) rules:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(z : D \triangleright \kappa : \Theta, M_0 : A, M_1 : B \quad x : A \triangleright \zeta : \Gamma \quad y : B \triangleright \xi : \Delta)</td>
</tr>
<tr>
<td>(z : D \triangleright \kappa : \Theta, \zeta[x := \text{casel } (M_0 \circ M_1)] = \zeta[x := M_0] : \Gamma)</td>
</tr>
</tbody>
</table>

(3.1)

| \(z : D \triangleright \kappa : \Theta, M_0 : A, M_1 : B \quad x : A \triangleright \zeta : \Gamma \quad y : B \triangleright \xi : \Delta\) |
| \(z : D \triangleright \kappa : \Theta, \xi[y := \text{caser } (M_0 \circ M_1)] = \xi[y := M_1] : \Delta\) |
Let \( q : D \to \Theta \bullet A \bullet B \), \( m : A \to \Gamma \) and \( n : B \to \Delta \). Then to satisfy the above equations in context we need that the following diagram commutes:

\[
\begin{array}{ccc}
D & \xrightarrow{q} & \Theta \bullet A \bullet B \\
\downarrow{\varphi} & & \downarrow{id_{\bullet A \bullet B}} \\
\Theta \bullet A \circ B & \xrightarrow{id_{\Theta \bullet A \circ B}} & \Theta \bullet A \bullet B
\end{array}
\]

We make the assumption that the above decomposition is unique. Moreover, supposing \( \Theta \) empty and \( m = id_A, n = id_B, q = id_A \bullet id_B = id_{A \bullet B} \) we obtain \((id_A \bullet id_B; PAR)^* = id_{A \bullet B} \) and similarly \((id_{A \bullet B})^*; PAR = id_{A \bullet B} \); hence we may conclude that there is a natural isomorphism

\[
\frac{D \to \Gamma \bullet A \bullet B}{D \to \Gamma \bullet A \circ B}
\]

so we can identify \( \bullet \) and \( \varphi \). Finally we see that the following \( \eta \) equation in context are also satisfied:

<table>
<thead>
<tr>
<th>( z : D \triangleright \kappa : \Upsilon, M : A \circ B )</th>
<th>( \varphi - \eta ) rule:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x : A \triangleright x : A )</td>
<td>( y : B \triangleright y : B )</td>
</tr>
<tr>
<td>( z : D \triangleright \kappa : \Upsilon, \text{case1}(M) )</td>
<td>( \varphi \text{ case2}(M) = M : A \circ B )</td>
</tr>
</tbody>
</table>

(3.2)

### 3.3. Linear subtraction

#### 3.3.1. Subtraction introduction

The introduction rule for subtraction has the form

\[
\frac{x : D \triangleright \kappa : \Gamma, M : A \quad y : B \triangleright \zeta : \Delta \quad x : D \triangleright \kappa : \Gamma, \zeta[y := y(M)] : \Delta, mkc(M, y) : A \setminus B \quad \varphi - \eta}{z : D \triangleright \kappa : \Gamma, \text{case1}(M) \varphi \text{ case2}(M) = M : A \circ B}
\]

This suggests a natural transformation with components

\[
\Phi_{D, \Gamma, \Delta} : C(D, \Gamma \bullet A) \times C(B, \Delta) \to C(D, \Gamma \bullet \Delta \bullet A \setminus B)
\]

natural in \( D, \Gamma, \Delta \). Taking morphisms \( e : D \to \Gamma \bullet A, f : B \to \Delta \) and \( a : D' \to D, c : \Gamma \to \Gamma', d : \Delta \to \Delta' \), by naturality we have

\[
\Phi_{D', \Gamma', \Delta'}((a; e; c \bullet id_A), (f; d)) = a; \Phi_{D, \Gamma, \Delta}(e, f); c \bullet d \bullet id_{A \setminus B}; \text{Join}(\Gamma', \Delta', A \setminus B)
\]

In particular, taking \( a = id_D, c = id_{\Gamma}, d : B \to \Delta \) and \( f = id_B \) we have:

\[
\Phi_{D, \Gamma, \Delta}(e, d) = \Phi_{D, \Gamma}(e, id_B); id_{\Gamma} \bullet d \bullet id_{A \setminus B}; \text{Join}(\Gamma, \Delta, A \setminus B)
\]

Writing \( MKC^B_{D, \Gamma}(e) \) for \( \Phi_{D, \Gamma}(e, id_B) \), \( \Phi_{D, \Gamma, \Delta}(e, d) \) can be expressed as the composition

\[
MKC^B_{D, \Gamma}(e); id_{\Gamma} \bullet d \bullet id_{A \setminus B}
\]

where \( MKC^B_{D, \Gamma} \) is a natural transformation with components

\[
MKC^B_{D, \Gamma} : C(D, \Gamma \bullet A) \times C(B, B) \to C(D, \Gamma \bullet B \bullet A \setminus B)
\]

so we make the definition

\[
[x : D \triangleright \kappa : \Gamma, \zeta[y := y(M)], mkc(M, y) : A \setminus B] = df
\]

\[
MKC^B_{D, \Gamma}[x : D \triangleright \kappa : \Gamma, M : A]; id_{\Gamma} \bullet [y : B \triangleright \zeta : \Delta] \bullet id_{A \setminus B}; \text{Join}(\Gamma, \Delta, A \setminus B)
\]
3.3.2. Subtraction elimination. The subtraction elimination rule has the form
\[
x : D \triangleright \kappa : \Gamma, M : A \setminus B \quad y : A \triangleright \xi : \Delta, N : B
\]
x : D \triangleright \text{postp}(y \mapsto N, M), \kappa : \Gamma, \xi[y := y(M)] : \Delta \setminus E
\]
This suggests a natural transformation with components
\[
\Psi_{D,\Gamma,\Delta} : \mathbb{C}(D, \Gamma \bullet (A \setminus B)) \times \mathbb{C}(A, \Delta \bullet B) \to \mathbb{C}(D, 1 \bullet \Gamma \bullet \Delta)
\]
natural in \(D, \Gamma, \Delta\). Given \(e : D \to \Gamma \bullet (A \setminus B)\), \(f : A \to \Delta \bullet B\) and also \(a : D' \to D\), \(c : \Gamma \to \Gamma'\), \(d : \Delta \to \Delta'\), naturality yields
\[
\Psi_{D',\Gamma',\Delta'}((a; e; c \bullet \text{id}_{A \setminus B}); (f; d \bullet \text{id}_B)) = a; \Psi_{D,\Gamma,\Delta}(e, f); c \bullet d; \text{Join}(\Gamma', \Delta')
\]
In particular, taking \(a : D \to \Gamma \bullet (A \setminus B)\), \(e = \text{id}_{\Gamma \bullet (A \setminus B)}\), \(c = \text{id}_\Gamma\), \(d : \Delta \to \Delta\), we obtain
\[
\Psi_{D,\Gamma,\Delta}(a, f) = a; \Psi_{D,\Gamma,\Delta}(\text{id}_{\Gamma \bullet (A \setminus B)}, f); \text{Join}(\Gamma, \Delta)
\]
Writing \(\text{POSTP}(f)\) for \(\Phi_{D,\Gamma,\Delta}(\text{id}_\Gamma \bullet (A \setminus B), f)\) we define
\[
[x : D \triangleright \text{postp}(y \mapsto N, M), \kappa : \Gamma, \xi[y := Y(M)] = df] \\
[x : D \triangleright \kappa : \Gamma, M : A \setminus B]; \text{id}_A \bullet \text{POSTP}[y : A \triangleright \xi : \Delta, N : B]; \text{Join}(\Gamma, \Delta)
\]
3.3.3. Equations in context. We have equations in context of the form

\[
\begin{array}{c}
x : D \triangleright \overline{N} : \Gamma, M : A \\
y : B \triangleright \overline{\Delta} : \Delta \\
z : A \triangleright \overline{L} : \Lambda, L : B
\end{array}
\]
\[
x : D \triangleright [\overline{N} : \Delta, \text{red}\overline{L} : \Lambda] = [\overline{N}[y := L[z := M]], L[z := M]]
\]  
(3.3)

where \(\overline{N} = \overline{N}[y := Y(M)]\), \(\text{red} = \text{postp}(z \mapsto L, \text{mkc}(M, Y))\) and \(\overline{L} = L[z := \text{mkc}(M, Y)]\).

Given morphisms \(n : D \to \Gamma \bullet A\) and \(m : A \to \Delta \bullet B\), for these equations to be satisfied we need the following diagram to commute:

\[
\begin{array}{c}
D \\
\text{MKC}^\beta(n)
\end{array} \xymatrix{ \Gamma \bullet (A \setminus B) \bullet B \ar[r]^-{\text{POSTP}(m) \bullet \text{id}_B} & \Gamma \bullet \Delta \bullet B }
\]

in particular, taking \(n = \text{id}_A\) we have

\[
\begin{array}{c}
A \\
\text{MKC}^\beta(\text{id}_A)
\end{array} \xymatrix{ (A \setminus B) \bullet B \ar[r]^-{\text{POSTP}(m) \bullet \text{id}_B} & \Delta \bullet B }
\]
Assuming the above decomposition to be unique, we can show that the \( \eta \) equation in context is also satisfied:

\[
\begin{array}{c}
z : D \triangleright N : \Delta, M : A \setminus B \\
x : A \triangleright x : A \\
y : B \triangleright y : B \\
z : D \triangleright N : \Delta, [\text{mkc}(X(M), Y) : A \setminus B, \text{postp}(x \mapsto Y(x), M)] = M : A \setminus B \end{array}
\]

(3.4)

and conclude that there is a natural isomorphism between the maps

\[
\begin{array}{c}
A \to \Delta \bullet B \\
A \setminus B \to \Delta
\end{array}
\]

i.e., that \( \setminus \) is the left adjoint to the bifunctor \( \bullet \).

3.4. Unit. 3.4.1 Unit rules. The introduction and elimination rules for the unit \( \bot \) are

\[
\begin{array}{c}
\bot \text{ introduction} \\
x : D \triangleright \kappa : \Gamma \\
x : D \triangleright \kappa : \Gamma, \text{connect to}(R) : \bot
\end{array}
\]

\[
\begin{array}{c}
\bot \text{ elimination} \\
x : \bot \triangleright \text{postp}(x)
\end{array}
\]

where \( R \in \kappa \).

The elimination rule is interpreted by a unique map \( \{\} : \bot \to 1 \).

The introduction rule requires a natural transformation with components

\[
\Phi_{D, \Gamma} : \mathbb{C}(D, \Gamma) \to \mathbb{C}(D, \Gamma \bullet \bot)
\]

natural in \( D \) and \( \Gamma \). Given morphisms \( e : D \to \Gamma \), \( d : D' \to D \) and \( c : \Gamma \to \Gamma' \), naturality yields

\[
\Phi_{D', \Gamma'}(d; e; c) = d; \Phi_{D, \Gamma}(e); c.
\]

Letting \( d : D \to \Gamma \) and \( e = id_\Gamma \), \( c = id_{\Gamma \bullet \bot} \) we have

\[
\Phi_{D, \Gamma}(d) = d; \text{Bot}_\Gamma
\]

where we write \( \text{Bot}_\Gamma \) for \( \Phi_\Gamma(id_\Gamma) \). We define

\[
[ x : D \triangleright \kappa : \Gamma, \text{connect to}(x) : \bot ] =_{df} [ x : D \triangleright \kappa : \Gamma ]; \text{Bot}_\Gamma.
\]

3.4.2. Equations in context. We may assume the operation \( \text{Bot}_\Gamma \) to be compatible with the generalized associativity and commutativity properties of \( \bullet \), so that for \( \Gamma = C_1, \ldots, C_n \) we have

\[
\Phi_{C_1, \ldots, C_i, \bot, \ldots, C_n}(id_\Gamma) : C_1 \bullet \cdots \bullet C_i \bullet \bot, \cdots \bullet C_n = \Phi_\Gamma(id_\Gamma) : C_1 \bullet \cdots \bullet C_n \bullet \bot
\]

for all \( i \leq n \). Together with naturality of \( \text{Bot}_\Gamma \) these yield the equations in context

\[
\begin{array}{c}
x : D \triangleright \kappa : \Gamma \\
x : D \triangleright \kappa : \Gamma, [\text{connect to}(R_i) = \text{connect to}(R_j)] \\
R_i, R_j \in \kappa
\end{array}
\]

\[
\begin{array}{c}
y : E \triangleright \zeta : \Gamma' \quad (R_i \in \kappa) \\
y : E \triangleright \zeta, [\text{connect to}(R_i) = \text{connect to}(R_j)] \\
R_i, R_j \in \zeta
\end{array}
\]

(3.5)
that correspond to the rewiring properties of $\bot$-links in the proof-net representation by [14, 15]. Moreover the equation in context

$$\begin{array}{c}
\bot - \beta \text{ rule} \\
x : D \triangleright \kappa : \Gamma \\
y : \bot \triangleright \text{postp}(y) \\
(R \in \kappa)
\end{array} \quad (R \in \kappa)$$

requires that for any $m : D \to \Gamma$ the following diagram commutes:

$$\begin{array}{c}
D \\
m \downarrow \\
\downarrow \iota
\end{array} \quad \begin{array}{c}
\bot \\
\text{id}_A \cdot \langle \rangle \\
\lambda_A \\
\Gamma
\end{array}$$

Assuming that this decomposition is unique and taking $m = id_A$ we have that $\text{Bot}_A \cdot \langle \rangle ; \lambda_A = id_A$. Arguing as before, we see that there is a natural isomorphism

$$\frac{D \to \Gamma \cdot 1}{D \to \Gamma \cdot 1}$$

(so we identify $\bot$ and 1) and that the following equation in context is satisfied:

$$\begin{array}{c}
\bot - \eta \text{ rule:} \\
\frac{z : D \triangleright \kappa : \Gamma, M : \bot}{x : \bot \triangleright \text{postp}(x)}
\end{array}$$

Let $\mathcal{L}$ be the signature having

- the types given by the following grammar on a collection of ground types $\gamma$:

$$A ::= \gamma \mid \bot \mid A \triangleright A \mid A \setminus A$$

- a collection of sorted function symbols including $\text{connect to}(-)$, $\text{postp}(-)$, $\varphi(-,-)$, $\text{casel}(-)$, $\text{caser}(-)$, $\text{mkc}(-,-)$, $\text{postp}(-,-)$.

We have proved the following

**Theorem 3.4.** Let $\mathcal{T} = (\mathcal{L}, A)$ be a theory with signature $\mathcal{L}$ having as axioms the equations in context in Table 4 and in (3.1) - (3.7). Let $(\mathbb{C}, \bullet, 1, \wedge, \wedge, \lambda, \rho, \gamma)$ be a symmetric monoidal left-closed category and $M$ a structure for $\mathcal{L}$ in $\mathbb{C}$. Then $M$ satisfies the equations in $\mathcal{A}$.

Moreover, define the syntactic category as the category $\mathcal{C}$ which has the formulas of multiplicative co-intuitionistic linear logic as objects and typed terms of the form $x : E \triangleleft \kappa : \Gamma$ (modulo renaming of the variable $x$) as morphisms. Set $x : E \triangleleft \kappa : \Gamma = y : E \triangleleft \zeta : \Gamma$ iff $\kappa = \zeta[y := x]$ is derivable from equations in context in Table 4 and in (3.1) - (3.7). Then we have

**Theorem 3.5.** The syntactic category is a symmetric monoidal left-closed category.

From this fact the categorical completeness theorem follows.
4. Extension to co-intuitionistic linear logic with coproducts and exponential

Let $\mathcal{L}^\oplus$ be $\mathcal{L}$ extended with additive disjunction $\oplus$ and the familiar functions $\text{inl} : A \to A \oplus B$, $\text{inr} : B \to A \oplus B$ and $\text{case} : A \oplus B \times (A \to C) \times (B \to C) \to C$. Then it is easy to extend the above result to show that if $\mathcal{C}$ has also the structure of coproducts, then a structure for $\mathcal{L}^\oplus$ in $\mathcal{C}$ satisfies also the theory $\mathcal{T}^\oplus$ where $\mathcal{A}$ is extended with familiar equations in context for $\text{inl}$, $\text{inr}$ and $\text{case}$. We shall not pursue this extension here.

The extension of $\mathcal{T}$ to a theory with the exponential $\oplus$ (why not?) is less simple. On one hand, one can dualize Benton, Bierman, De Paiva, Hyland’s definition of a linear category $[11, 13]$ and obtain in this way a sound and complete categorical semantics for co-intuitionistic linear logic. The construction of weakly distributive categories with storage operators based on proof-nets by Blute, Cockett and Seely $[14]$ provides a categorical model for both exponentials $!$ and $\otimes$. On the other hand, the semantics for the exponential $!$ can recovered in the context of Nick Benton’s treatment of Linear Non Linear logic $[12]$. After dualizing the linear part of $\text{LNL}$ one should be able to recover the semantics for $\otimes$ and at the same time obtain a framework where the duality of intuitionistic and co-intuitionistic logic can be studied. We leave the development of this approach to future work and focus on the categorical semantics of the multiplicative and exponential $\otimes$ fragment of co-intuitionistic linear logic.


We begin by dualizing the definition of a linear category $[11, 13]$.

**Definition 4.1.** A dual linear category $\mathcal{C}$ consists of

1. A symmetric monoidal left-closed category together with
2. a symmetric co-monoidal monad $(?, \eta, \mu, \mathbb{n}_{-\otimes}, \mathbb{n}_{\otimes})$ (namely, the functor $\otimes$ is co-monoidal with respect to $\otimes$ and the linear transformation $\eta, \mu$ are co-monoidal) such that

   (i) - each free $\otimes$-algebra $(\otimes A, \mu A)$ carries naturally the structure of a commutative $\otimes$-monoid (i.e., for each $(\otimes A, \mu A)$ there are distinguished monoidal natural transformations $i_A : \otimes \to \otimes A$ and $c_A : \otimes A \otimes \otimes \to \otimes A$ which form a commutative monoid and are algebra morphisms);

   (ii) - whenever $f : (\otimes A, \mu A) \to (\otimes B, \mu B)$ is a morphism of free algebras, then it is also a monoid morphism.

**Remarks 4.2.** By Maietti, Maneggi de Paiva and Ritter (see $[27]$, Prop. 25), condition 2(ii) is equivalent to the requirement that $\mu$ is a monoidal morphism.

i) To say that the functor $\otimes$ is symmetric co-monoidal means that it comes equipped with a comparison natural transformation $\mathbb{n}_{A,B} : (\otimes A \otimes B) \to \otimes A \otimes B$ and a morphism $\mathbb{n}_{\otimes} : \otimes \to \otimes$, satisfying

\[
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array{c}}
CO-INTUITIONISTIC LINEAR LOGIC

\((A \varphi B) \varphi C\) \xrightarrow{\eta_{A,B}} \vdash A \varphi B \varphi ? C \xrightarrow{\eta_{A,B}} \vdash A \varphi ? C

\((A \varphi B) \varphi ? C\) \xrightarrow{\eta_{A,B}} \vdash A \varphi ? C \xrightarrow{\eta_{A,B}} \vdash A \varphi ? C

\((A \varphi (B \varphi C)) \xrightarrow{\eta_{A,B}} \vdash A \varphi ? (B \varphi C) \xrightarrow{id_{A \varphi B} \varphi_{A,B,C}} \vdash A \varphi ? (B \varphi C)

\((A \varphi B) \xrightarrow{\eta_{A,B}} \vdash A \varphi ? B \quad \text{and naturality:} \quad (A \varphi B) \xrightarrow{\eta_{A,B}} \vdash A \varphi ? B

\((B \varphi A) \xrightarrow{\eta_{A,B}} \vdash B \varphi ? A \quad \text{and naturality:} \quad (A \varphi B) \xrightarrow{\eta_{A,B}} \vdash A \varphi ? B

(ii) To say that \(\eta\) and \(\mu\) are co-monoidal is to say that the following diagrams commute:

\[ \eta_{A,B} \quad \vdash A \varphi ? B \quad \text{and} \quad \mu_{A,B} \quad \vdash A \varphi ? B

\[(?A \varphi B) \xrightarrow{\eta_{A,B}} \vdash A \varphi ? B \quad \text{and} \quad \mu_{A,B} \quad \vdash A \varphi ? B

(iii) To say that the natural transformations \(i_A : \bot \rightarrow A\) and \(c_A : A \varphi ? A \rightarrow A\) are monoidal means that they are compatible with the comparison maps, i.e., that the following diagrams commute:

\[ i\bot \quad \vdash \bot \quad \text{and} \quad \lambda = p \quad \vdash \bot \quad \text{and} \quad \lambda = p \quad \vdash \bot \quad \text{and} \quad \lambda = p \quad \vdash \bot \quad \text{and} \quad \lambda = p \quad \vdash \bot

\[ (A \varphi A) \varphi (B \varphi B) \xrightarrow{c_{A \varphi B}} \vdash A \varphi ? B

where \(iso\) is the canonical isomorphism derived from symmetry and associativity;
(iv) Finally for the free algebra morphisms to be monoid morphism we require that the following diagrams commute:

\[ \begin{array}{ccc}
\vdash \alpha & \rightarrow & \vdash \\
\mu_A & \rightarrow & \mu_A \\
\vdash \mu_A & \rightarrow & \vdash \\
?A \rightarrow & ?A \\
\end{array} \]  
and  

\[ \begin{array}{ccc}
\vdash \alpha & \rightarrow & \vdash \\
\mu_A & \rightarrow & \mu_A \\
\vdash \mu_A & \rightarrow & \vdash \\
?A \rightarrow & ?A \\
\end{array} \]

4.2. Term and equations in context. To sketch a proof that a dual linear category is a model of co-intuitionistic linear logic with storage operator \( \otimes \) we give the term in context and the equation in context relevant to the dereliction, weakening, contraction and storage rules. These conditions are dual to those in Figures 4.1-4.5 in G. M. Bierman’s thesis [13], pp. 112-142. Since in our context the exponential rules for dereliction, weakening and contraction do not involve \texttt{let} constructions, some of these conditions result immediately from properties of substitution. There are three Equations in Context expressing “\( \beta \) reductions” for the storage operator in Table 6. Finally there are Categorical Equations in Context in Table 7.

The key decision, discussed at length in G. M. Bierman’s thesis [13] pp. 127-131, arises in the analysis of the Dereliction-Storage reduction given by the equation in context in Table 6. By repeating for the rules of dereliction and storage the kind of analysis done for \( \texttt{par} \),
Dereliction - Storage:
\[
v : E \vdash \kappa : \Gamma, M : C \quad x : C \triangleright \overline{Q} \mid \overline{N} : ?\Delta \\
v : E \vdash \kappa : \Gamma, [\text{store}(\overline{Q}, \overline{N}, \overline{y}, x, [M]) \mid \overline{y}(x([M])) : ?\Delta = \overline{Q}[x := M] \mid \overline{N}[x := M] : ?\Delta]
\]

Contraction - Storage:
\[
v : E \vdash \kappa : \Gamma, M_0 : ?C, M_1 : ?C \quad x : C \triangleright \overline{Q} \mid \overline{N} : ?\Delta \\
v : E \vdash \kappa : \Gamma, [\text{store}(\overline{Q}, \overline{N}, \overline{y}, x, [M_0, M_1]) \mid \overline{y}(x([M_0, M_1])) : ?\Delta = \text{store}(\overline{Q}, \overline{N}, \overline{y}, x, M_0, M_1) \mid (\overline{y}(x(M_0)), \overline{y}(x(M_1))) : ?\Delta]
\]
where \( ?\Delta = ?D_1, \ldots, ?D_m \) and \((\overline{y}(x(M_0)), \overline{y}(x(M_1))) : ?\Delta \) stands for \([y_1(x(M_0)), y_1(x(M_1))], \ldots, [y_m(x(M_0)), y_m(x(M_1))] : ?D_1, \ldots, ?D_m \)

Weakening - Storage:
\[
v : E \vdash \kappa : \Gamma \quad x : C \triangleright \overline{Q} \mid \overline{N} : ?\Delta \quad R \in \kappa \\
v : E \vdash \kappa : \Gamma, [\text{store}(\overline{Q}, \overline{N}, \overline{y}, x, \text{connect to}(R)) \mid \overline{y}(x(\text{connect to}(R))) : ?\Delta = (\overline{\text{connect to}(R)}) : ?\Delta]
\]
where \( ?\Delta = ?D_1, \ldots, ?D_m \) and \((\overline{\text{connect to}(R)}) : ?\Delta \) stands for \(\text{connect to}(R) : ?D_1, \ldots, \text{connect to}(R) : ?D_m \)

Table 6: Equations in context for the \( ? \) storage operator

\[\begin{array}{c}
\text{Subtraction} \quad \text{and} \quad \text{Unit, we see that in order to model the storage rule we need a natural transformation } \Phi_{E, \Gamma} : C(E, \Gamma \bullet ?A) \times C(A, ?\Delta) \to C(E, \Gamma \bullet ?\Delta). \text{ By naturality considerations this is given by its action } \Phi_{\Gamma}(id_{\Gamma \bullet ?A}, d) = \eta_f \circ d^* \text{ on morphisms } d : A \to ?\Delta. \text{ Similarly, for the dereliction rule we need a natural transformation } \Psi : C(_A, A) \to C(_A, ?A) \text{ and by applying Yoneda’s Lemma we see that its action is given by a morphism } \eta_A : A \to ?A.
\end{array}\]

We can certainly define a functor \( ? : C(A, \Gamma) \to C(?A, ?\Gamma) \) by \( f \mapsto (f; \eta_f)^* \). Now by the equation in context for dereliction-storage we have that following the diagram commutes:

\[
\begin{array}{ccc}
??A & \xrightarrow{(\eta_A)^*} & ??A \\
\downarrow \eta_A & & \downarrow \eta_A \\
??A & \xrightarrow{?A} & ??A \\
\end{array}
\]

Assuming the above decomposition to be unique, we have \((\eta_A)^* = id_{??A}\) and thus the derivations

\[
\eta_A : x : ?A \triangleright [x] : ??A \quad \text{and} \quad ?\eta_A : z : ?A \triangleright \text{store}([[x]], y, x, z) \mid y(x(z)) : ??A
\]

must be identified. Now it can be shown that identifying \( \eta_A \) and \( ?\eta_A \) forces the functor \( ? \) to be idempotent: \( ??f = ?f \). In order to avoid such collapse, the functor \( ? \) is only assumed to be a K modality, and the properties of S4 are given by the natural transformations
Monad:
\[ z : A \triangleright \text{store}(x, y, x, z) \mid y'(x'(z)) : ?A = x : ?A \]

Algebra 1
\[
\begin{align*}
v : E \triangleright \kappa : \Gamma, M : ?C & \quad x : C \triangleright \overline{P} \mid N : ?\Delta \\
v : E \triangleright \kappa : \Gamma, P[x := x(M)], & \quad \text{store}(\langle N, \text{connect to}(R), (y, M) \mid y(x(M)) : ?\Delta, y(x(M)) : ?A) =} \\
& \quad = \text{store}(\langle N, y, M \mid y(x(M)) : ?\Delta, \text{connect to}(R') : ?A) \\
\end{align*}
\]
where \( R \in P \cup N \) and \( R' \in P[x := x(M)] \cup \overline{y}(x(M)) \)

Algebra 2
\[
\begin{align*}
v : E \triangleright \kappa : \Gamma, M : ?C & \quad x : C \triangleright \overline{P} \mid N : ?\Delta, N_0 : ?A, N_1 : ?A \\
v : E \triangleright \kappa : \Gamma, P[x := x(M)], & \quad \text{store}(\langle N, N_0, N_1 \rangle, (y, y_0, y_1, x, M) \mid \overline{y}(x(M)) : ?\Delta, [y_0(x(M)), y_1(x(M))] : ?A) =} \\
& \quad = \text{store}(\langle N, N_0, N_1 \rangle, (y, y_0, y_1, x, M) \mid \overline{y}(x(M)) : ?\Delta, [y_0(x(M)), y_1(x(M))] : ?A) \\
\end{align*}
\]

Monoid 1
\[
\begin{align*}
v : E \triangleright \kappa : \Gamma, M : ?C & \quad R \in \kappa \cup M \\
v : E \triangleright \kappa : \Gamma, [\text{connect to}(R) : ?C = M : ?C] \\
\end{align*}
\]

Monoid 2
\[
\begin{align*}
v : E \triangleright \kappa : \Gamma, M : ?C & \quad R \in \kappa \cup M \\
v : E \triangleright \kappa : \Gamma, [[\text{connect to}(R), M] : ?C = M : ?C] \\
\end{align*}
\]

Monoid 3
\[
\begin{align*}
v : E \triangleright \kappa : \Gamma, M_0 : ?C, M_1 : ?C & \\
v : E \triangleright \kappa : \Gamma, [[M_0, M_1] : ?C = [M_1, M_0] : ?C] \\
\end{align*}
\]

Monoid 4
\[
\begin{align*}
v : E \triangleright \kappa : \Gamma, M_0 : ?C, M_1 : ?C, M_2 : ?C & \\
v : E \triangleright \kappa : \Gamma, [[[M_0, M_1], M_2] : ?C = [M_0, [M_1, M_2]] : ?C] \\
\end{align*}
\]

Table 7: Categorical Equations in Context

\( \eta : A \to ?A \) and \( \mu : ?A \to ?A \) of the monad \((?, \eta, \mu)\). Here \( \mu_A \) is given by the proof
\[ z : ?A \triangleright \text{store}(x, y, x, z) \mid y'(x'(z)) : ?A \]
and the commutative diagram required by the definition of a monad

\[
\begin{array}{ccc}
?A & \xrightarrow{\mu_A} & ?A \\
\downarrow{\eta_A} & & \downarrow{\eta_A} \\
?A & \xrightarrow{id_A} & ?A
\end{array}
\]

identifies \( id_A : x : ?A \triangleright x : ?A \) with the following derivation \( ?A; \mu_A \):
The normal form of the derivation $\eta A; \mu A$ is the following one:

$$z : ?A \ni x : ?\mu A \ni x : {A \multimap [x]} : ?A \ni z' : ?A \multimap z' : ?A \ni x' : ?A \multimap x' : ?A$$

where $t = y(x(z)) : ?A$

as in the Categorical Equation in Context for Monad of Table 7. Further details are left to the reader.

5. Conclusion.

In order to provide a categorical semantics for co-intuitionistic logic - given that as remarked by Tristan Crolard [18] co-exponents in the category Set are trivial - we have given a categorical semantics for intuitionistic multiplicative and exponential co-intuitionistic linear logic, from which our desired results follows by dualizing J-Y. Girard’s embedding of intuitionistic logic into intuitionistic linear logic.

In this task we started from a term assignment to multiplicative co-intuitionistic logic, which has been proposed as an abstract distributed calculus dualizing the linear $\lambda$ calculus [2, 3, 7]: in our view such dualization underlies the translation of the linear $\lambda$-calculus into the $\pi$-calculus (see [10]). Our dual distributed calculus is itself a restriction to a co-intuitionistic consequence relation of Crolard’s term assignment to subtraction in the framework of the $\lambda\mu$-calculus: to subtraction introduction and elimination rules and to their $\beta$ reduction global operations of binding and global substitution are assigned; these operations may appear as notationally awkward at first sight but are forced on us by the removal of the $\mu$-rule and of the $\mu$-variable abstraction used in Crolard’s approach. A computational application of our notation is suggested in the proof of the Decomposition Property (Proposition 2.2). Since the dependencies of a variable $y : C$ from $n$ binders make – coroutine and postpone are represented in a term $y(t_1(\ldots t_n(M)\ldots)) : C$ by the terms $t_i : D_i$, this representation can be used to compute the assignment of a probabilistic event $C$ to such a term according to the assignments $D_i$ to the terms $t_i : D_i$ and to prove that probabilities are preserved from the premise to the disjunction of the conclusions in a multiplicative co-intuitionistic derivation.

Our work required a lengthy exercise on well-known results by Benton, Bierman, Hyland and de Paiva [11, 13], with the considerable help given by Blute, Cockett, Seely and Trimble’s work [14, 15]. To assess the merits and advantages of our work we need to evaluate the syntax for the exponential rules: here again the storage rule may appear notationally quite heavy, but it is a straightforward implementation of the act of storing. On the other hand the advantages of working in the dual system are completely evident in the treatment of dereliction and contraction, where the awkward let operations and related naturality conditions are replaced by simple operations on lists. Finally, the treatment of weakening is also completely standard, thanks also to Blute, Cockett, Seely and Trimble’s work [14, 15] on the notion of rewiring.
References


APPENDIX A. Example

Consider the following computation in the simply typed λ-calculus. We write $N$ for $(A \supset A) \supset (A \supset A)$ and $N^\perp$ for $(C \setminus C) \setminus (C \setminus C)$.

$\lambda h ^{A \supset A} . (\lambda f ^{A \supset A} . \lambda x ^{A} . ffx)(\lambda y ^{A \supset A} . \lambda y ^{A} . gyy)h : N \rightsquigarrow _\beta (i)$

$\lambda h ^{A \supset A} . (\lambda f ^{A \supset A} . \lambda x ^{A} . ffx)(\lambda y ^{A} . hhy) : N \rightsquigarrow _\beta (ii)$

$\lambda h ^{A \supset A} . \lambda x ^{A} . (\lambda y ^{A} . hhy)(\lambda y ^{A} . hhy)x : N \rightsquigarrow _\beta (iii)$

$\lambda h ^{A \supset A} . \lambda x ^{A} . ((\lambda y ^{A} . hhy)hhy)x : N \rightsquigarrow _\beta (iv)$

$\lambda h ^{A \supset A} . \lambda x ^{A} . hhxh : N \rightsquigarrow _\beta (v)$

In Table 8 we give a Natural Deduction derivation in “tree form” with the assignment of the term (ii) $\vdash \lambda h . (\lambda f . \lambda x . ffx)(\lambda y . hhy) : N$. In Table 9 we consider a Natural Deduction derivation of $n^\perp : N$ in the subtraction only fragment of co-intuitionistic logic; such a derivation is exactly dual of that in Table 8: it is also in “tree form”, in fact it yields the same tree as in Table 8 read from bottom up. Its term assignment, with the p-terms $P_1, P_2, P_3, P_4$ in the conclusion, belong to a dual calculus defined in [1] and briefly described here, with the property that to each β reduction in the simply typed λ calculus there corresponds a set of rewritings of the computational context in the dual calculus and a reduction sequence $t_1, t_2, \ldots$ of simply typed λ terms terminates if and only if the dual sequence $t_1^\perp, t_2^\perp, \ldots$ terminates (see [1], section 6.1). The derivation in Table 10 results from that in Table 9 by one step of normalization.

The grammar of the dual calculus for the subtraction-only fragment of co-intuitionistic is as follows (see [1], section 6, definition 10):

$M ::= x \mid x(M) \mid \text{mkc}(M, x)$

$\ell ::= [] \mid [M_1, \ldots, M_n]$ for some $n$.

$P ::= \text{postp}(y \mapsto \ell[y := y(M)], M)$. 
Table 8: Natural Deduction tree for $(ii) \vdash \lambda h. (\lambda f. \lambda x. ff x)(\lambda y. hhy) : N$

Here non-empty lists $\ell$ are flattened and occur only within p-terms. A notion of term expansion allows to define the substitution of a flat list for a free variable in (a flat list of) terms, yielding a flat list. In this way we take care of contraction of discharged conclusions resulting from a subtraction elimination. A conclusion introduced by weakening is assigned an empty list. To decorate natural deduction trees we use a notation which ignores the distinction between local and remote binding, as discussed in Note 1.7 (iii). Notice that the grammar of the dual calculus used in this section is actually a fragment of the grammar of the linear dual calculus presented in this paper.  

Next we translate the co-intuitionistic natural deduction derivations of Tables 9 and 10 into co-intuitionistic linear logic. In Tables 11 and 12 we adapt the graphical notation of Tables 9 and 10 to our linear calculus and notice that that the derivation in Table 12 results from that in Table 11 by applying two normalization steps, a subtraction reduction followed by a storage - contraction reduction. The graphical notation should help to catch a glimpse of the reduction process more vividly. Here we present the Natural Deduction derivation of Table 11 in the sequent-style typing judgements of our official calculus.

\footnote{We have not explored the possibility of assigning the empty list to the formulas $?C$ introduced by weakening also in the linear calculus; this would distinguish the case of weakening from that of the $\bot$-introduction rule, where terms of the form connect to ($R$) would still be used.}
(2) : \( P_2 = \text{postp}(g \mapsto [m_2, m_1], m_3) \)  
(1) : \( P_1 = \text{postp}(b \mapsto [x], g) \)  
(3) : \( P_3 = \text{postp}(d \mapsto [y], j) \)  
(4) : \( P_4 = \text{postp}(e \mapsto [m_5, m_4], n) \)

\[ m_1 = \text{mkc}(a, x) : C \setminus C \]
\[ x = x(a) : C \]
\[ m_2 = \text{mkc}(b, a) : C \setminus C \]
\[ a = a(b) : C \]
\[ m_3 = \text{mkc}(e, j) : N^\perp \]
\[ d = d(j) : C \]
\[ j = j(e) : C \]
\[ e = e(n) : C \setminus C \]
\[ m_5 = \text{mkc}(c, y) : C \setminus C \]
\[ y = y(c) : C \]
\[ c = c(d) : C \]
\[ m_4 = \text{mkc}(c, y) : C \setminus C \]
\[ y = y(c) : C \]
\[ c = c(d) : C \]

**Table 9:** co-IL tree for the dual of (ii) \( \vdash \lambda h.(\lambda f.\lambda x.f f x)(\lambda g.h h y) : N \)

**Sequent-style Natural Deduction.** (i) The derivation \( D \) corresponding to the graph inside box \( B_{k,M} \) of Table 11 is as follows.

\[ \vdash d : C \mid d : C \mid c : C \mid c : C \]
\[ \vdash d : C \mid \text{mkc}(d, c) : C \setminus C \]
\[ \vdash d : C \mid m_5 : C \setminus C \]
\[ \vdash d : C \mid m_4 : C \setminus C \]
\[ k : C \setminus C \]
\[ k : C \setminus C \]
\[ k : C \setminus C \]
\[ k : C \setminus C \]
\[ \vdash d : C \mid [m_5] : ?(C \setminus C) \]
\[ \vdash d : C \mid [m_4] : ?(C \setminus C) \]

Applying the ?-E rule with major premise \( j : ?(C \setminus C) \mid j : ?(C \setminus C) \) we obtain a derivation of the following sequent:

\( j : ?(C \setminus C) \mid \text{store}(P_3, M, y_0, k, j) \mid y_0(k(j)) : ?(C \setminus C) \).
(1): $P_1' = \text{postp}(b \mapsto [x], e)$
(3'): $P_3' = \text{postp}(d' \mapsto [y'], m_2)$

(4): $P_4' = \text{postp}(e \mapsto [m_5', m_5'', m_4', m_4''], n)$
(3''): $P_3'' = \text{postp}(d'' \mapsto [y''], m_1)$

Redex

Finally, by applying subtraction introduction to it with the axiom $e : C \setminus C \triangleright e : C \setminus C$ we obtain a derivation $D_1$ of the following sequent:

$$e : C \setminus C \triangleright \text{store}(P_3, M, y_0, k, j(e)) \mid y_0(k(j(e))) : ?(C \setminus C), \mkc(e, j) : N^\perp$$

where $N^\perp = (C \setminus C) \\triangleright ?(C \setminus C)$.

(ii) By applying the same steps as in derivation $D$, but relabelling of the terms, we obtain a derivation $D_0$ of the following sequent:

$$g : C \setminus C \triangleright \text{postp}(b \mapsto x, g) \mid [[m_1][m_2][b := b(g)] : ?(C \setminus C) \mid m_1 = \mkc(b, a), m_2 = \mkc(a, x), b = b(g), a = a(b) \text{ and } x = x(a).$$

(iii) Now if we apply $D_1$ to $D_0$ with the formula $\mkc(e, j) : N^\perp$ as major premise of subtraction elimination then we obtain a derivation $D^+$ ending with following sequent:

$$e : C \setminus C \triangleright \text{Store} \text{ postp}(g \mapsto M_0, \mkc(e, j)) \mid y_0 : ?(C \setminus C)$$
The pair introduction / elimination inferences determines the only Redex in $D^+$. A final subtraction elimination with the axiom $n : N^\perp \triangleright n : N^\perp$ concludes the derivation in our example.
(1) \( P_1 = \text{postp}(b \mapsto x, e) \)

\[ y_2 = y_0(k_2) \]

\[ k_2 = \text{mk}(m_2) \]

\[ B_{k_2, M_2} \]

---

\( (4') \quad P_4 = \text{postp}(e \mapsto [[y_2][y_1]], n) \)

\[ y_2 = y_0(k_2) \]

\[ k_1 = \text{mk}(m_1) \]

\[ B_{k_1, M_1} \]

---

Table 12: **linear co-IL** analysis of the proof in Table 10